NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PROPOSAL TO DEVELOP ENHANCEMENTS AND EXTENSIONS OF FORMAL MODELS FOR RISK ASSESSMENT IN SOFTWARE PROJECTS

by

Michael R. Murrah

September 2002

Thesis Advisor:	Luqi
Second Reader:	Nabendu Chaki

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Proposal to Develop Enhancements and Extensions of Formal Models For Risk Assessment in Software Projects			**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael R. Murrah			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

Over the past 40 years limited progress has been made to help practitioners estimate the risk and the required effort necessary to deliver software solutions. Recent developments improve this outlook. Researchers from the Naval Postgraduate School developed a formal model for risk assessment used to estimate software project risk. This model is based on easily obtainable software metrics quantifiable early in the software development process.

The previous risk assessment model was developed on data collected from a series of experiments conducted on the Vite'Project simulation. This unique approach provided a starting point towards a proven formal model for risk assessment, one that can be applied early in the software development lifecycle. Approaching software risk estimation has never previously been successfully accomplished in this manner.

This research provides definitive evidence that software risk assessment can be conducted early in software development using quantifiable metrics and simple techniques. Previous models have been enhanced based on calibrations against post-mortem projects. These enhancements result from many threads of research; extension of input metrics, increased simulations, simulations calibrated on actual projects, and model development. The research delivers an improved risk assessment model, one that has been validated against thousands of post-mortem projects, with applicability on any software development activity.

14. SUBJECT TERMS Risk Assessment, Formal Models, Software Estimation Models, Software Metrics, Project Management, Monte Carlo Simulation			**15. NUMBER OF PAGES** 143
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PROPOSAL TO DEVELOP ENHANCEMENTS AND EXTENSIONS OF FORMAL MODELS FOR RISK ASSESSMENT IN SOFTWARE PROJECTS

Michael R. Murrah
Major, United States Army
B.S., Georgia College and State University, 1993
M.S., University of Missouri-Rolla, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Michael R. Murrah

Approved by: Luqi
Thesis Advisor

Nabendu Chaki
Second Reader

Chris Eagle
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Over the past 40 years limited progress has been made to help practitioners estimate the risk and the required effort necessary to deliver software solutions. Recent developments improve this outlook. Researchers from the Naval Postgraduate School developed a formal model for risk assessment used to estimate software project risk. This model is based on easily obtainable software metrics quantifiable early in the software development process.

The risk assessment model was developed on data collected from a series of experiments conducted on the Vite'Project simulation. This unique approach provided a starting point towards a proven formal model for risk assessment, one that can be applied early in the software development lifecycle. Software risk estimation has previously enjoyed minimal success in this manner.

This research provides definitive evidence that software risk assessment can be conducted early in software development using quantifiable metrics and simple techniques. Extensions are made possible based on calibrations against post-mortem projects. These enhancements result from many threads of research; extension of input metrics, increased simulations, simulations calibrated on actual projects, and model development. The research proposes an improved risk assessment model, one that has been validated against thousands of post-mortem projects, with applicability on any software development activity.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Acknowledgements often go overlooked but enough cannot be said to and about the people instrumental in writing and researching a thesis. I knew not what lay ahead as I embarked on this mission. I am forever grateful to the many people who believed in the possibility of the accomplishment.

I certainly have to thank my primary advisor, Professor Luqi, who got me interested in this topic early after my arrival to the Naval Postgraduate School. Without her constant prodding, I probably would still be looking for a thesis topic. The list of support she provided is endless and a simple word of thanks is not enough.

Secondly, multiple displays of encouragement were always available from Dr. Man-Tak Shing and Prof. Richard Riehle, two people who make the Software Engineering Program first class. Dr. Shing, with his never-ending knowledge of Prototype System Description Language (PSDL) and Computer Aided Prototyping System (CAPS) was always a constant source of reference and Prof. Riehle with his never-ending knowledge of software engineering took the time to push me through this demanding program.

I saved the best for last. No words can express the gratitude I have towards my immediate family Lesley and Reba. Their undying support and encouragement sustained me through times when I doubted myself. Balance is always essential to maintain forward momentum, and you have always been instrumental to maximizing both work and family. Without you, this would not have been possible. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

A. INTRODUCTION

The current state of the art techniques of risk assessment primarily rely on checklists and human expertise. This constitutes a weak approach because different people could arrive at different conclusions from the same scenario. The difficulty of estimating the duration of projects applying evolutionary software processes adds intricacy to the risk assessment problem.

1. Nogueira's Risk Assessment Model

Nogueira's research introduces a formal method to assess the risk and the duration of software projects automatically, based on measurements that can be obtained early in the development process. The method has been designed according to the characteristics of evolutionary software processes, and utilizes quantifiable indicators such as efficiency, requirement volatility and complexity. The formal model, based on these three indicators estimates the duration and risk of evolutionary software processes. The approach introduces benefits in two fields:

- Automation of risk assessment
- Early estimation methods for evolutionary software processes

Nogueira developed four software risk estimation models that show great promise in determining a software projects' associated risk early in the software development life cycle. The models accomplish early estimation by utilizing a set of quantifiable metrics that can be collected from the beginning of project development. In actuality, the requirements volatility metric is an estimation during the first development cycle and during subsequent development cycles is quantifiable. After each iteration of software development, the required input metrics can be applied to the model in order to reduce the error in the model's results.

The minimum required input metrics, to support risk assessment, required for Nogueira's estimation model are the following:

a. **Efficiency (EF)** – The efficiency of the organization can be measured observing the fit between people and their roles. Nogueira's research indicates that the efficiency of an organization can be directly calculated by computing the ratio of direct time (working and correcting errors) divided by the idle time (time spent without work to do).

b. **Requirements Volatility (RV)** – Requirements volatility expresses how difficult the requirement elicitation process is. The requirements volatility is obtained by the following formula

Requirements Volatility = Birth Rate Percentage + Death Rate Percentage

Birth Rate Percentage (BR%) = the percentage of new requirements incorporated in each cycle of the software evolution process as calculated by:

BR% = (New Requirements / Total Requirements) * 100 percent

Death Rate Percentage (DR%) = the percentage of requirements that are dropped by the customer in each cycle of the evolution process as calculated by:

DR% = (Deleted Requirements / Total Requirements) * 100 percent

c. **Complexity (CX)** – Complexity has a direct impact on quality because the likelihood that a component fails is directly related to its complexity. The complexity metrics can be determined in two forms: large granular complexity and fine granular complexity. These two forms of complexity can be directly determined from software specifications written in the Prototype System Description Language (PSDL) [BerzLuqi90].

Large Granular Complexity (LGC) expresses the relational complexity of the system as a function of the number of operators (O), data streams (D), and types (T)

LGC = O + D + T

Fine Granular Complexity (FGC) expresses the relational complexity of each operator in the system and is a function of the fan-in and fan-out data streams related to the operator. For the purposes of the completed research and our notion of future

research, the FGC metric is too specialized; our efforts concentrate on just the representation of the LGC.

FGC = fan-in + fan-out

Software developers can utilize Nogueira's four models to assess either the development time required to develop a project or determine the associated probability of completing a software project given the project's duration.

2. Previous Validation Research

In this section of the paper we present the results of validation attempts when using Nogueira's estimation models. The first is a result of the research conducted by Nogueira in his initial research and supplies data from simulations and comparisons to one project. The second validation endeavor is the results of research conducted on two additional projects [JohnPirr01].

2.1. Nogueira's Validation

In conducting his research, Nogueira derived some initial conclusions with the models. The simulations showed that the three risk factors observed during the causal analysis (efficiency, requirements volatility, and complexity) have compound effects over the three parameters of the Weibull distribution.

Nogueira illustrates the results of the models against 16 simulated projects. Each model derives an increasing degree of accuracy based on: metrics from the three risk factors, Weibull cumulative density function, and the derivation of the time.

Models 1-2.

Model 1 can be used when the requirements volatility is small. Model 2 considers the three factors (EF, RV, and CX), but neglects the combined effect of EF and RV. Figure 1 illustrates the results of the models which were calculated using 95% of confidence (p=0.95). Note the errors as vertical segments between the estimated and real values.

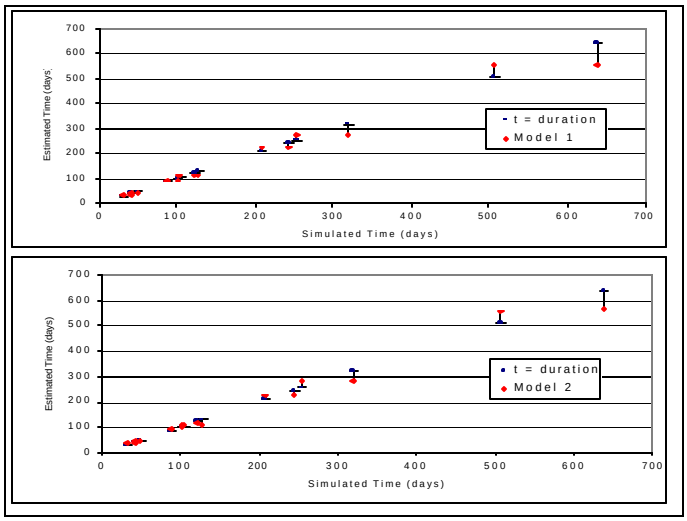


Figure 1 Scatter Plot of Models 1-2.

Model 3. Model 3, illustrated in Figure 2, considers the three factors as well as the combined effects of EF and RV. The analysis of variance shows that the samples obtained from the simulations and the samples obtained from the estimates using Model 1, 2 or 3 cannot be statistically differentiated.

Another interesting result is that the errors remain in the range of ±15% for all of the scenarios. This result is interesting if we compare it with the results of COCOMO (±20% in the best cases). Barry Boehm in reference to the validation of COCOMO said, "In terms of our criterion of being able to estimate within 20% of projects actuals, Basic COCOMO accomplishes this with only 25% of the time, Intermediate COCOMO 68% of the time, and Detailed COCOMO 70% of the time." [Boeh81]

Model 4. Model 4, Figure 2, can be used for any range of complexity and requirements volatility, and considers the three factors, their combined effects, and the following a priori assumptions:

- A project with 0 LGC will take 0 days
- a , b , and g > 0
- If RV increases the p(x<=t) decreases
- If CX increases then p(x<=t) decreases

- If EF increases then p(x<=t) increases

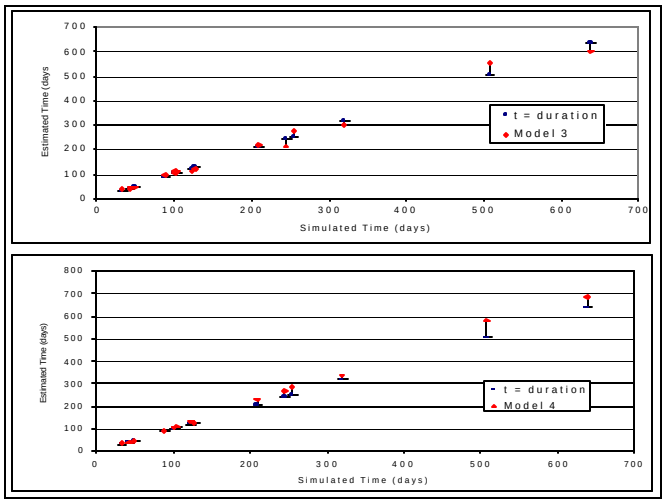


Figure 2 Scatter Plot of Models 3-4.

The scatter plot in Figure 2 compares the simulated times versus the estimated times. Most of the errors are overestimations and the duration of the project has no effect over the percentage of error. Model 4 is conservative. The maximum overestimation error was less than 16% and the maximum underestimation was less than 4%.

Model 4 gives a good estimation for projects between 4,000 and 20,000 LGC (128 and 640 KLOC of Ada). The estimation seems to be too optimistic for projects smaller than 1000 LGC but it is quite good for larger projects. To verify the model Nogueira used a real project consisting of 1836 LGC developed in 1.5 years by the Uruguayan Navy[1]. Model 4 predicts 17 months instead of 18 months, the actual development time.

2.2. Additional Project Validation

Project A. We used Nogueira's Model 4 to calculate the probability of completion curve for the projects. For consistency, we used working days, defined as 22 days per month, the same as used in the original Nogueira model.

[1] SIMTAS a simulator for war gaming with 75,240 lines of code.

The model predicted that the minimum time, in days, necessary to have a probability of completion of 100% is approximately 260 working days. When compared to the actual time it took, which was 336 working days, the model predicted completion sooner. The model predicted 76 working days less, or a 22.6% delta.

(1 - (260 / 336)) (100) = 22.6.

At this point, with 22.6% variability, we decided to investigate and see what the original estimated completion date was from project records. The original estimation was 200 working days, with the project schedule slipping 136 working days for build 3. The developer missed the original completion estimation by 40.5%.

(1- (200 / 336)) (100) = 40.5.

The Nogueira model missed the developer's original estimate by 23.1%

(1-(200 / 260)) (100) = 23.1

Does this mean that the Nogueira model is too optimistic as are most developers' estimates, or is it a better fit? This data point leaves us with an inconclusive position as to the validation of the model against the first project. It appears that there is a difference when using real projects with real data versus simulated project data, and this reflects what the real world is – unpredictable.

Project B. We used Nogueira's Model 4 to calculate the probability of completion curve for Build 2 using; BR=2.59, DR=3.04, RV=5.63, O=2544, D=4010, T=1003. The model predicted Impossible.

Actual time for build 2 took from 4/24/00 until 7/10/00 or 68 working days at 22 working days a month. We believe this inconsistency is due primarily because the calculation for the LGC count is based on all six Computer Software Configuration Items (CSCI). Core functionality on three CSCIs; CSCI-A, CSCI-B, and CSCI-C had been previously developed and validated. However, the builds during this period, involved addition of functionality to the following CSCIs: CSCI-D, CSCI-E, and CSCI-F. That is, build 2 was modifying only a portion of the total software system code, but the LGC data gives a view of all six CSCIs combined.

The available data was not broken down into separate CSCIs, nor does it, post-mortem, identify the code that was being worked in a previous software release. We cannot fault the developer for not collecting metrics for research concepts that they are not aware of, nor do we believe that this type of data collection is a requirement of CMM level 3.

A finding of this research is the need to adjust the CX when applying the Nogueira model to evolved projects that are developing or enhancing only a portion of their CSCIs.

Additionally, this project did not utilize a lower case tool such as Rational Rose. We believe use of such a tool is essential when attempting to apply the Nogueira formal model, as it provides the capability to collect detailed information, over the software development lifecycle, that can later be extracted and used for input to the Nogueira model metrics.

3. Issues with Nogueira's Risk Assessment Model

Applying Nogueira's risk assessment model, in its current form, presents a number of issues that must be resolved before substantial progress can be achieved validating the model's results. The first issue and most notable draw back when using Nogueira's risk assessment model is limited confidence that the model provides valid results. This is due to three factors: the limited amount of time that the model has been in existence, the model has not been exercised on a wide base of real world projects (completed or on-going), and the fact that the model was developed using simulation techniques. The first factor noted can only be dealt with in the passage of time. However, this research will exploit a unique opportunity to impact the latter two issues.

Although Nogueira's research shows promise in estimating the associated risk when developing software systems, the model has not been significantly exercised beyond theoretical simulation. Three "real world" projects to date have been applied against the estimation model [JohnPiir01]. It should be noted that all three of these projects were exercised post-mortem. Model validity has not been demonstrated in the context targeted by the model's original design, estimating risk early in a software project's life cycle.

A second issue that exist when using Nogueira's risk assessment model is the required input metrics. This issue is a double-edged sword. A major attraction to using Nogueira's model are these metrics. They are determined in a definitive, quantifiable manner and can be derived extremely early in the software development process [NoguLuqiBhat00]. However, these metrics are quite unique. Currently, outside of the academic environment, it is not common practice to collect these unique metrics in the required form to utilize Nogueira's risk assessment model.

In order to establish confidence in the usefulness and accuracy of Nogueira's risk estimation model, the model must be exercised against numerous projects. It would be ideal, and perhaps over time, to exercise the model according to it original design; early in the software development cycle. However, the next logical step is to continue to exercise the model in a post-mortem basis. Before this can be accomplished, two things need to happen: First correlations must be determined between Nogueira's required metrics and metrics that are frequently collected in historical project databases. By establishing metrics correlations, the model can be exercised against an additional project base helping address the second factor of problem one. And second, a method other than the use of PSDL to generate O, D and T metrics counts must be developed. Nogueira's model was based on using PSDL to automatically scan and generate counts for O, D, and T input to his model. It is unlikely that PSDL was used on any programs that we have post-mortem data on.

The final problem associated with Nogueira's risk assessment model is the configuration of the Vite'Project simulation. Nogueira developed the configuration of Vite'Project using Organizational Consultant expert system. Fictitious software engineering organizations were developed to represent the typical software development department. Based on the results of establishing fictitious CMM level 2 and level 3 organizations, the Vite'Project was calibrated. Calibrating the simulation in this manner, could yield different results than calibrating the simulation with actual information derived from real projects. If Nogueira's model can be verified by reprogramming the Vite'Project configuration this would provide additional assessment to the third factor of problem one.

4. Research Outline

The proposed research will expand the efforts of the previous validation effort. Figure 3 outlines the research approach.

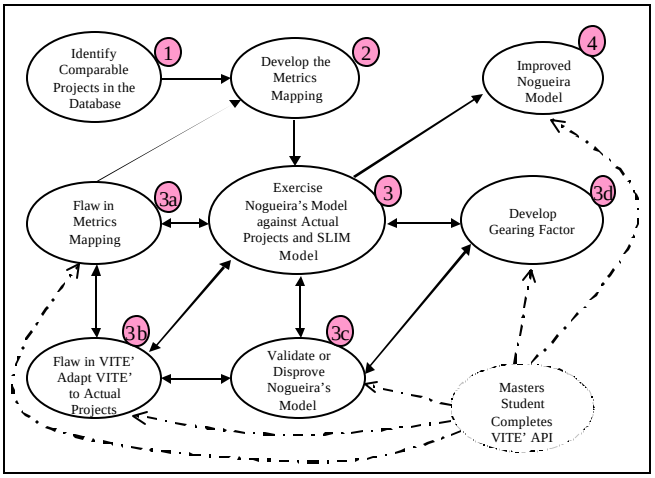


Figure 3 Phases of Research.

Phase one: During phase one of the research, post-mortem projects will be identified whose characteristics are similar to the characteristics of the three projects previously exercised against Nogueira's risk assessment model. This affords the opportunity to begin with a baseline before proceeding to future phases.

Phase two: This is the most challenging phase of the research and we hypothesize that this phase will consume the majority of the available resources. In this phase, detailed analysis is conducted against the available metrics that have been collected on the projects established during phase one. Correlations are determined in the available data against the three metrics that are necessary when utilizing Nogueira's model. Upon completion of this phase, when a suitable "metric map" has been developed, research can continue to phase three. The intent of the metric map is to provide a common platform to exercise Nogueira's model using metrics that were not originally collected for this purpose.

Phase three: Once a suitable metric map has been established, research continues by exercising Nogueira's model against the set of post-mortem projects determined in phase one. This phase is essential to establish confidence in the results

produced when using Nogueira's model. Additionally during this phase, another risk assessment method is introduced, Quantitative Software Management's® (QSM) SLIM, to help in the validation process. Essentially, there will be a comparison of three artifacts: the recorded project performance, the estimated project performance using Nogueira's model, and the estimated project performance as determined by QSM's SLIM. An assumption during this phase will be the accuracy of QSM's SLIM. Of course, if the expected results are not achieved during this phase, additional research must be performed to determine the cause of the variance.

Phase three (a): One potential cause of the variance observed during phase three could be a flaw in the metric map determined during phase two. Continued research will be conducted to modify the mapping and eventually minimize the chance that the metric map is the source of the deviation.

Phase three (b): Another factor that can influence deviation between the actual project data, Nogueira's estimation model, and QSM's SLIM estimation model is the original configuration used to establish project scenarios in the Vite'Project. Organizational Consultant expert system was used to establish fictitious software engineering organizations. Research may indicate that reprogramming the Vite'Project with actual information from software development organizations could yield different results in the Vite'Project simulation. This was a fundamental factor in the development of Nogueira's research. A substantial change in the simulated results could require extensive rework of Nogueira's model.

Phase three (c): Finally, after exhausting Phases three (a & b), research may lead to examination of Nogueira's model with closer scrutiny. If deviation continues to present itself when conducting phase three, we may have essentially resort to "ground zero" to establish potential conflicts. *It should be noted that phases three (a, b, & c) should not be considered mutually exclusive. Research could indicate that partial modifications are required in all three sub-phases.*

Phase three (d): Nogueira's risk assessment model is perfectly suited for any evolutionary software process because it follows the same philosophy. Nogueira presents

no hypothesis of the model's validity when the model is exercised outside of this domain. Once phase three is accomplished and confidence has been established against the set of projects determined during phase one, the model can be exercised against additional projects, from different industry sectors and different software development methodologies. This may require the development of what we are calling a "gearing factor". In this research, the use of this term is intended to represent a value that is multiplied by the results determined in Nogueira's model, adjusting the results for the new domain. In some cases the model may provide suitable results without the use of a gearing factor, other domains and development methodologies may require this adjustment due to the unique nature of the software's development.

Phase four: Phase four of the proposed research is the culmination of all of the proposed research. This phase delivers the improved Nogueira model. A caveat to this phase and all of the sub-phases conducted during phase three is the introduction of the Vite'Project API. This automated tool will improve the statistical significance obtained when utilizing the Vite'Project simulation, greatly increasing the number of simulation runs provided by the simulation.

5. Validation

We propose to validate our research by conducting controlled experiments against post-mortem projects. QSM, founded in 1978 by Larry Putnam, has collected and maintained an extensive database of over 5,000 software projects [SLIM01]. Experiments can be conducted, utilizing the available software metrics from QSM's database, that correlate the required metrics in Nogueira's model. This will afford our research the means to evaluate actual projects against Nogueira's model.

Another source of validation is obtained by configuring Vite'Project with actual software project development information. As previously mentioned, Vite'Project scenario's were originally established by the creation of fictitious software development organizations. Different results could be derived from simulations configured according to actual projects.

Finally, we propose to increase the statistical significance of Nogueira's software risk assessment model. We can accomplish this by increasing the simulation runs of each scenario through automation via the Vite' API when available.

6. Conclusion

This research introduces a research plan to validate a formal risk assessment model for software projects based on probabilities and metrics automatically collectable early in the project. The approach enables a project manager to evaluate the probability of success of the project very early in the life cycle. For more than twenty years the estimation standards (COCOMO 81, COCOMO II, Putnam) have been characterized by a common limitation: the requirements should be frozen in order to make estimations. This promising model removes this important limitation, facing the reality that requirements are inherently variable.

The problem of risk assessment for projects has been treated as unstructured. Research shows, and experiments will prove, a structured method to solve the problem based on metrics automatically collected from the project baselines. This contribution impacts the software engineering state of the art, as well as risk management in general. These metrics measure three risk factors identified in the research: complexity, requirements volatility, and efficiency. The subjectivity issue characteristic of previous research has been eliminated. Any decision-maker will arrive at the same estimates, independently of his or her expertise.

Finally, current research is based on simulations and a small set of real projects. It is desirable to collect and analyze metrics and completion times of a larger set of real software projects to confirm and refine the models. Our research will provide the missing elements from the models, validation, enhancements, and extensions.

I. INTRODUCTION

Over the past 40 years limited progress has been made to help practitioners estimate the risk and the required effort necessary to deliver software solutions. Recent developments improve this outlook, one in particular, the research conducted by Juan Carlos Nogueira. Dr. Nogueira developed a formal model for risk assessment that can be used to estimate a software project's risk when examined against a desired development time-line. This model is based on easily obtainable software metrics. These metrics are quantifiable early in the software development process.

Nogueira developed his model based on data collected from a series of experiments conducted on the Vite'Project simulation. This unique approach provides a starting point towards a proven formal model for risk assessment, one that can be applied early in the software development lifecycle. Approaching software risk estimation has never previously been successfully accomplished in this manner.

The proposed research will provide definitive evidence that software risk assessment can be conducted early in software development using quantifiable metrics and simple techniques. Enhancements will be made to Nogueira's model, based on calibrations against post-mortem projects. These enhancements will result from many threads of research; extension of input metrics, increased number of simulation runs, simulation scenarios based on actual projects, and the introduction of a "gearing factor". Ultimately, the research will yield a risk assessment model that has been validated against thousands of post-mortem projects, having application in any software development activity.

A. THE PROBLEM

The first problem and most notable draw back when using Nogueira's risk assessment model is limited confidence that the model provides valid results. This is due to three factors: the limited amount of time that the model has been in existence, the model has not been exercised on a wide base of real world projects (completed or on-going), and the fact that the model was developed using simulation techniques. The first

factor noted can only be dealt with in the passage of time. However, we have the unique opportunity to impact the latter two issues.

Although Nogueira's research shows promise in estimating the associated risk when developing software systems, his model has not been significantly exercised beyond theoretical simulation. Three "real world" projects to date have been applied against the estimation model. It should be noted that all three of these projects were exercised post-mortem. Model validity has not been demonstrated in the context targeted by its original design, estimating risk early in software project's life cycle.

A second problem in validating Nogueira's risk assessment model is the required input metrics. This problem presents itself as a double-edged sword. A major attraction to using Nogueira's model is these metrics. They are determined in a definitive, quantifiable manner and can be derived extremely early in the software development process. However, these metrics are quite unique. Currently, outside of the academic environment, it is not common practice to collect these unique metrics in the required form to utilize Nogueira's risk assessment model.

In order to establish confidence in the usefulness and accuracy of Nogueira's risk estimation model, the model must be exercised against numerous projects. It would be ideal, and perhaps over time, to exercise the model according to it original design; early in the software development cycle. However, the next logical step is to continue to exercise the model in a post-mortem basis. Before this can be accomplished, correlations must be determined between Nogueira's required metrics and metrics that are frequently collected in historical project databases. By establishing metrics correlations, the model can be exercised against an additional project base helping address the second factor of problem one.

The final problem associated with Nogueira's risk assessment model is the configuration of the Vite'Project simulation. Nogueira developed the configuration of Vite'Project using Organizational Consultant expert system. Fictitious software engineering organizations were developed to represent the typical software development organization. Based on the results of establishing fictitious CMM level 2 and level 3 organizations, the Vite'Project was calibrated. Calibrating the simulation in this manner

could yield different results than calibrating the simulation with actual information derived from real projects. If Nogueira's model can be verified by reprogramming the Vite'Project configuration this would provide additional assessment to the third factor of problem one.

B. RESEARCH QUESTIONS

- How do the metrics required for using Nogueira's risk assessment model correlate to metrics collected in real world projects?
- How does Nogueira's risk assessment model perform when exercised on projects whose characteristics and domain are similar to the three projects already evaluated?
- How does Nogueira's risk assessment model perform when compared against commercially available estimation tools on post-mortem projects?
- What is the impact on Nogueira's risk assessment model if metrics collected on actual projects are programmed into the original Vite'Project simulation?
- What are the necessary enhancements, if any, required to evolve Nogueira's risk assessment model to provide valid results when exercised on real projects?
- Is an introduction of a "gearing factor" necessary to obtain valid results when expanding Nogueira's risk assessment model into other software domains?

C. GENERAL RESEARCH DESIGN

Figure 4 below outlines the intended approach to accomplish the proposed research. Phase one: During phase one of the research, post-mortem projects will be identified whose characteristics are similar to the characteristics of the three projects previously exercised against Nogueira's risk assessment model. This affords the opportunity to begin with a baseline before proceeding to future phases.

Phase two: This is the most challenging phase of the research and I hypothesize that this phase will consume the majority of the available resources. In this phase, detailed analysis is conducted against the available metrics that have been collected on the projects established during phase one. Correlations are determined in the available data against the three metrics that are necessary when utilizing Nogueira's model. Upon completion of this phase, when a suitable "metric map" has been developed, research can

continue to phase three. The intent of the metric map is to provide a common platform to exercise Nogueira's model using metrics that were not originally collected for this purpose.

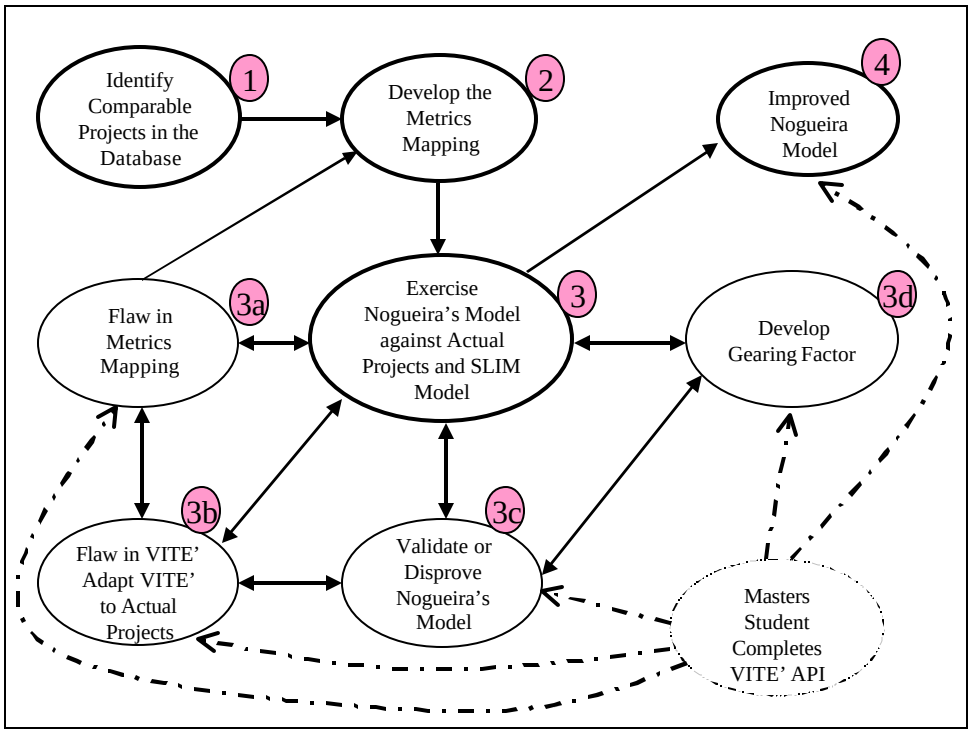


Figure 4 Phases of Research.

Phase three: Once a suitable metric map has been established, research continues by exercising Nogueira's model against the set of post-mortem projects determined in phase one. This phase is essential to establish confidence in the results produced when using Nogueira's model. Additionally during this phase, another risk assessment method is introduced, Quantitative Software Management's® (QSM) SLIM, to help in the validation process. Essentially, there will be a comparison of three artifacts: the recorded project performance, the estimated project performance using Nogueira's model, and the estimated project performance as determined by QSM's SLIM. An assumption during this phase will be the accuracy of QSM's SLIM. Of course, if the expected results are not achieved during this phase, additional research must be performed to determine the cause of the variance.

Phase three (a): One potential cause of the variance observed during phase three could be a flaw in the metric map determined during phase two. Continued research will be conducted to modify the mapping and eventually minimize the chance that the metric map is the source of the deviation.

Phase three (b): Another factor that can influence deviation between the actual project data, Nogueira's estimation model, and QSM's SLIM estimation model is the original configuration used to establish project scenarios in the Vite'Project. As previously mentioned, Organizational Consultant expert system was used to establish fictitious software engineering organizations. Research may indicate that reprogramming the Vite'Project with actual information from software development organizations could yield different results in the Vite'Project simulation. This was a fundamental factor in the development of Nogueira's research. A substantial change in the simulated results could require extensive rework of Nogueira's model.

Phase three (c): Finally, after exhausting Phases three (a & b), research may lead to examination of Nogueira's model with closer scrutiny. If deviation continues to present itself when conducting phase three, I may have essentially resort to "ground zero" to establish potential conflicts.

It should be noted that phases three (a, b, & c) should not be considered mutually exclusive. Research could indicate that partial modifications are required in all three sub-phases.

Phase three (d): Nogueira's risk assessment model is perfectly suited for any evolutionary software process because it follows the same philosophy [Nogu00]. Nogueira presents no hypothesis of the model's validity when the model is exercised outside of this domain. Once phase three is accomplished and confidence has been established against the set of projects determined during phase one, the model can be exercised against additional projects, from different industry sectors and different software development methodologies. This may require the development of what I am calling a "gearing factor". In this research, the use of this term is intended to represent a value that is multiplied by the results determined in Nogueira's model, adjusting the results for the new domain. In some cases the model may provide suitable results without

the use of a gearing factor, other domains and development methodologies may require this adjustment due to the unique nature of the software's development.

Phase four: Phase four of the proposed research is the culmination of all of the proposed research. This phase delivers the improved Nogueira model. A caveat to this phase and all of the sub-phases conducted during phase three is the introduction of the Vite'Project API. This automated tool will improve the statistical significance obtained when utilizing the Vite'Project simulation, greatly increasing the number of simulation runs provided by the simulation.

D. ORGANIZATION OF THESIS

Chapter II of this thesis presents the theoretical foundations for the proposed research. Chapter III describes the conceptual framework of the research. In Chapter IV, the proposed dissertation is demonstrated to be academically acceptable. This chapter includes the detailed approach to accomplishing the phases above. Chapter V presents some initial analysis on the software risk model.

Finally, Chapter VI leaves the reader with conclusions and directions for future research. Following the base chapters in the thesis, appendixes are included to detail the available metrics, a sampling of the original 112 projects identified in phase one, modified simulation results, references, etc.

II. THEORETICAL FOUNDATION

A. THEORETICAL FOUNDATION FOR RISK MANAGEMENT

1. Basis

This Thesis expands previous research on software project risk assessment, namely predicting the success of the project. The only ways to evaluate the degree of success of a project are (a) to compare planned and actual schedule, (b) to compare planned and actual costs; and (c) to compare planned and actual product characteristics. Software reliability, an emergent branch of software engineering, has addressed this last part. However, the first two issues have not yet been emphatically addressed.

For many years research has greatly increased our knowledge of software projects. Among such software laws, it is known that:

- Manpower and time are not interchangeable [Broo74].
- Human productivity rates are highly variable, and function and size are highly correlated with errors and duration of the project [Putn80].
- The majority and most costly errors are introduced during the requirements phase [Boeh81].
- Life-cycle manpower patterns follow tailed probability curves [Nord63], [Putn80, 92, 96, 97], [Boeh81].
- Standards, good practices, guidelines, and heuristics improve the development process [Hump89].

At present, there are Computer Aided Software Engineering (CASE) tools that can improve productivity. Also, Macro models can estimate, with different degrees of success, the effort and duration of software projects [Albr79], [Boeh81, 00], [Putn97]. What is not available is a model of the internal phenomenology of the software life cycle. Without the knowledge of such a model, scientific risk assessment is almost impossible. The software process is a set of activities with dependency relationships that occur over a certain period of time. From this point of view software development projects do not differ from any other type of project. At the beginning of such a process, a great deal of uncertainty exists. This uncertainty can be reduced through effort, which can be

expressed as time and cost. As time goes by, the level of uncertainty usually decreases because more information becomes available.

Unfortunately, the main resources (time and budget) are interrelated and must be traded off. So project managers, as decision-makers, must choose between making early decisions with a great deal of uncertainty, or postponing decisions by trading time for information. The concept of early measure is emphasized because recognizing the risks in the early phases of software development increases the probability of success, and therefore, improves the competitive advantage. The focus of this research is on automatically collectable measures since risk identification should be as objective as possible and not impose any substantial additional workload.

2. IEEE Standard for Software Life Cycle Processes

In March 2001, IEEE published IEEE Std 1540-2001, which may be used independently of any particular software life cycle process standard, or in conjunction with IEEE/EIA 12207.0-1996. Although IEEE/EIA 12207.0-1996 describes a standard process for the acquisition, supply, development, operations, and maintenance of software, this standard does not provide a process for risk management. IEEE Std 1540-2001 risk management standard provides that process.

The purpose of risk management is to identify and mitigate the risks continuously [IEEE01]. The risk management process is a continuous process for systematically addressing risk throughout the life cycle of a product or service. IEEE Std 1540-2001 describes this process consists of the following activities:

- Plan and implement risk management
- Mange the project risk profile
- Perform risk analysis
- Perform risk monitoring
- Perform risk treatment
- Evaluate the risk management process

Figure 5 illustrates the IEEE Std 1540-2001 risk management process. This model and the discussion following is an extract from the standard.

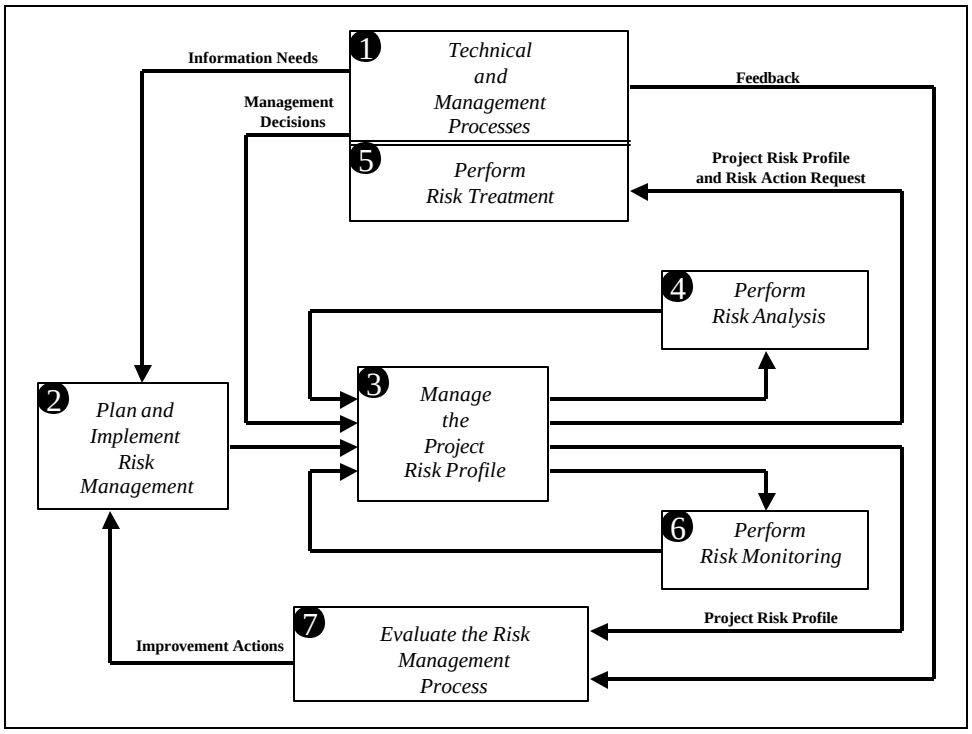


Figure 5 IEEE Risk Management Process Model.

Managerial and technical processes involving the stakeholders define the information requirements (i.e., the information the stakeholders require to make informed decision involving risks) the risk management process must support ❶. These information requirements are passed to both the "plan and implement risks management" and the "manage the project risk profile" activities. In the "plan and implement risk management" activity ❷, the policies regarding the general guidelines under which risk management will be conducted, the procedures to be used, the specific techniques to be applied, and so forth, are defined.

In the "manage the project risk profile" activity ❸, the current and historical risk management context and risk state information are captured. The project risk profile includes the sum total of all the individual risk profiles (i.e., the current and historical risk information concerning and individual risk), which, in turn, includes all the risk states.

The project risk profile information is continually updated and maintained through "perform risk analysis" activity ❹, which identifies the risks, determines their likelihood and consequences, determines their risk exposures, and prepares risk action requests recommending treatment for risks determined to be above their risk threshold(s).

Treatment recommendations, along with the status of other risks and their treatment status, are sent to management for review ❺. Management decides what risk treatment is implemented for any risk found to be unacceptable. Risk treatment plans are created for risks that require treatment. These plans are coordinated with other management plans and other ongoing activities.

All risk are continually monitored until they no longer need to be tracked during the "perform risk monitoring" activity ❻. In addition, new risks are sought out.

Periodic evaluation of the risk management process is required to ensure its effectiveness. During the "evaluate the risk management process" activity ❼, information, including user and other feedback, is captured for improving the process or for improving the organization's or project's ability to manage risk. Improvements defined as a result of evaluation are implemented in the "plan and implement risk management" activity ❷.

The software risk management process is applied continuously throughout the product life cycle. However, activities and tasks of the risk management process interact with the individual risks in an interactive manner once the risk management process begins. For example, in the "perform risk analysis" activity ❹, a risk may be re-estimated several times during the performance of risk evaluation due to an increase in knowledge about the risk gained during the evaluation task itself. The risk management process is not a "waterfall" process.

Although the IEEE Risk Management Process Model provides a framework for organizations to conduct risk management, it fails to address the basic research questions posed by the Nogueira research; identifying quantifiable early collectable metrics in the software process and utilizing these metrics to assess project risk.

Results of the proposed research will have applicability in the IEEE 1540-2001 standard. Section 5.1.3 Perform Risk Analysis describes activities that should be performed to conform to the standard [IEEE02]. These activities are listed below:

- Identify the initiating events, hazards, threats, or situations that create risks
- Estimate the likelihood of occurrence, the consequences for each risk, and the expected timing of the risk
- Evaluate each risk or defined combination of risks against its applicable threshold, generate alternatives to treat risks above their risk thresholds, and make recommendations for treatment based on a priority order.

The proposed research will provide enhanced models, which in their proven form will provide quantifiable metrics to identify risk and to estimate the impact of these risks on software projects.

B. ESTIMATION MODELS

1. Nogueira

Nogueira developed four software risk estimation models that show great promise at determining a software projects' associated risk early in the software development life cycle. The models accomplish early estimation by utilizing a set of quantifiable metrics that can be collected from the beginning of project development. In actuality, the requirements volatility metric is estimated during the first development cycle and during subsequent development cycles is quantifiable. After each iteration of software development, the required input metrics can be applied to the model in order to reduce the error in the model's results.

The minimum required input metrics, to support risk assessment, that are required for Nogueira's estimation model are the following:

a. Efficiency

The efficiency of the organization can be measured observing the fit between people and their roles [Nogu00]. Nogueira's research indicates that the efficiency of an organization can be directly calculated by computing the ratio of direct time (working and correcting errors) divided by the idle time (time spent without work to do).

b. Requirements Volatility

Requirements volatility expresses how difficult the requirement elicitation process is. The requirements volatility is obtained by the following formula [Nogu00].

Requirements Volatility = Birth Rate Percentage + Death Rate Percentage

Birth Rate Percentage (BR%) = the percentage of new requirements incorporated in each cycle of the software evolution process as calculated by:

BR% = (New Requirements / Total Requirements) * 100 percent

Death Rate Percentage (DR%) = the percentage of requirements that are dropped by the customer in each cycle of the evolution process as calculated by:

DR% = (Deleted Requirements / Total Requirements) * 100 percent

c. Complexity

Complexity has a direct impact on quality because the likelihood that a component fails is directly related to its complexity [Nogu00]. The complexity metrics can be determined in two forms: large granular complexity and fine granular complexity. These two forms of complexity can be directly determined from software specifications written in the Prototype System Description Language (PSDL) [Luqi90].

Large Granular Complexity (LGC) expresses the relational complexity of the system as a function of the number of operators (O), data streams (D), and types (T)

LGC = O + D + T

Fine Granular Complexity (FGC) expresses the relational complexity of each operator in the system and is a function of the fan-in and fan-out data streams related to the operator [Nogu00].

FGC = fan-in + fan-out

Nogueira's research serves as the foundational basis for the proposed research. As mentioned in Chapter I, Nogueira's models have their own set of problems associated with them. The rest of this dissertation proposal is presented to show the plan to overcome the shortcoming of Nogueira's model.

2. Putnam

World-renowned author, lecturer, and consultant on software productivity, quality, and lifecycle estimating, founded Qualitative Software Management, Inc. in

1978. He has written over 30 technical papers and four books on the subject. QSM is the most senior software management firm in the industry [QSMa].

QSM's methods, tools, and training provide pro-active solutions for software productivity measurement and improvement, cost and schedule estimating, size estimating, and runaway project prevention. Many of the world's major software producers are represented in the QSM database. Companies who use QSM's services and products come from a variety of industry: micro code development, real-time avionics and weapons systems, process control and systems software to large financial, banking and MIS systems.

QSM has dedicated over twenty years to collecting and analyzing data from software development projects. During this time, QSM has assembled detailed data on more than 5,000 software systems developed since 1980 [QSMb].

QSM developed and maintains software tools to help companies manage their software development effort. These tools are incorporated into what QSM calls the SLIM suite of tools. SLIM (Software Lifecycle Management) encapsulates the essence of QSM's service to the industry. Utilizing these tools, along with access to the projects stored in QSM's software project database, provides a unique opportunity to experiment with Nogueira's estimation models. The tools available from QSM are the following [QSMb]:

SLIM Metrics. SLIM Metrics provides the user a comprehensive and customizable platform to query, conduct statistical analysis, graphing, and reporting tools necessary to assess and compare the performance of your projects.

SLIM Control. SLIM Control is a management tool for project tracking, forecasting and presentation of software project performance while under development. The core measurement and comparisons include: schedule, effort, cost, staffing and reliability.

SLIM Estimate. SLIM Estimate is a management tool for estimation, analysis and presentation of software project characteristics. These characteristics include: schedule, effort and quality. Each estimate has an associated probability range.

SLIM Master Plan. SLIM Master Plan is an analysis tool that accepts time-series measurement data from multiple sources and provides a single-chart portrayal of each measurement, either by individual source or in aggregate.

3. Keshlaf and Hashim

Keshlaf and Hashim recognize there exist important weaknesses in the existing approaches for software risk estimation. Their research is based on the premise that risk documentation and concentrating on top risks are the best ways to save developers time and effort and produce good results in reducing software risks [Kesh00]. The authors have designed an improved software risk model and developed an implementation tool; called SoftRisk, based off the model. Figure 6 details the characteristics of the SoftRisk's model. Following Figure 6 is an extract of their research explaining the procedure.

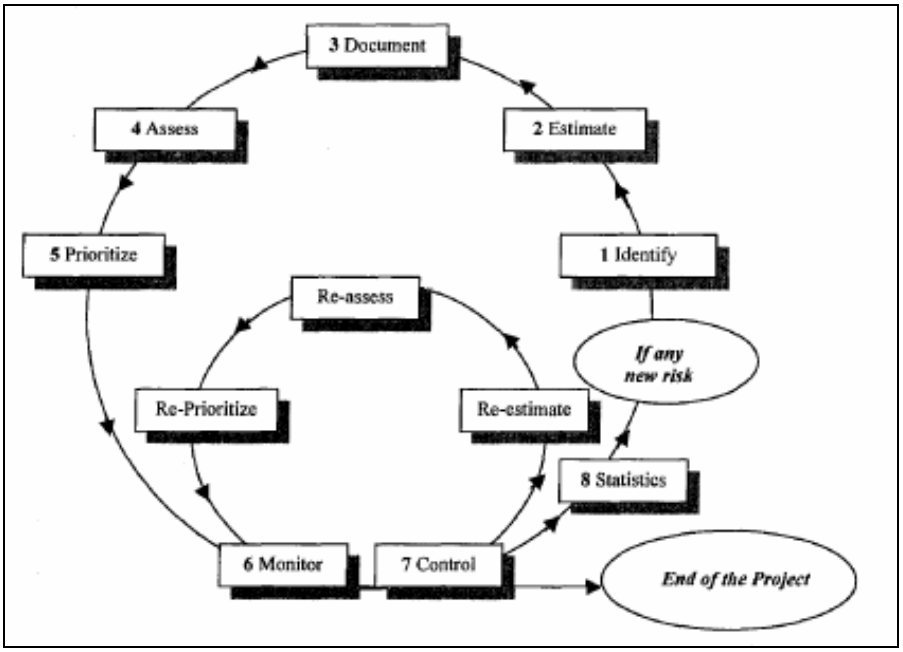


Figure 6 SoftRisk Model.

Step 1: Risk Identification: All potential risks must be identified. Two forms are suggested to be used, one for general risks data which can be used by any software development project, whereas, the second is for specific projects risks data which affect the desired project.

Step 2: Risk Probability and Magnitude Estimation: Each identified risk (Specific risk) is estimated in terms of its probability and magnitude. A special checklist can be used to assist in estimating both probabilities and magnitudes of risks. The estimation should be under one of the following categories:

1 – Negligible	2 – Very Low	3 - Medium
4 – High	5 – Very High	6 – Extra High

Step 3: Risk Documentation: All generic and specific risks data must be documented in order to use them for tracking project's situation, statistical operations and future risk predictions.

Step 4: Risk Assessment: Each risk must be assessed based on its probability and magnitude. It is advisable to use risk exposure (RE) formula:

RE = Risk Probability * Risk Magnitude

Step 5: Prioritization and Highlighting Highest Ten Risks: All risks are sorted based on RE values and the top ten risks for each inspection are prioritized and listed.

Step 6: Monitoring (Graphic Representation): A line graph is recommended to be used to represent RE values. The graph is divided into three zones: red for catastrophic, yellow for medium, and green for negligible risks.

Step 7: Controlling: Based on the severity of the risk, a suitable reduction technique is performed. It could be mitigation, contingency, or crisis plan. Re-estimation, reassessment, and action documentation must be carried out after performing any one of the reduction techniques.

Step 8: Performing Statistical Operation and Going Back to Step 1: Any suitable statistical operation can be carried out if need be. However, if the project is still ongoing it is highly recommended to continue performing risk management steps to avoid any problems.

SoftRisk requires the user to make qualitative estimates about the associated risk probability and magnitude estimation. The tool takes the qualitative inputs and converts this information into quantitative data for internal calculations. Approaching risk management in this fashion does not facilitate removing subjective inputs from the model. The basic problem of deriving quantitative metrics early in the project still exists.

The SoftRisk model does recognize the importance of documenting software development risks. It even introduces risk reduction advice and documents reduction actions. However, all of the model's results are derived from a subjective basis.

Finally, SoftRisk has not been thoroughly tested to establish its validity. The preliminary tests were conducted to establish accurate functionality of the tool itself, not testing the results of the tool. Additionally, the authors recognize the fact that the SoftRisk tool only prepares the risks data to become ready for future statistical operations.

4. Mitre Corporation

Software development activities can gain valuable information and insights from examining what other development activities have done to mitigate risk when developing software projects. The Mitre Corporation has developed a commercial tool called the Risk Assessment and Management Program (RAMP), which is a risk management information system that provides interactive support for identifying, analyzing, and sharing risk mitigation experience.

Operating on an Internet platform, RAMP lets authorized users access project-specific risk mitigation experiences across the corporation from locations anywhere in the world [Garv97]. RAMP contains a database of systems engineering-related project risks and mitigation strategies, along with more than 1,000 links to risk-relevant resources and contacts around the world.

According to [Garv97], RAMP provides users the ability to:

- Identify and collaborate with domain experts in specific risk and technology areas.
- Query experts via e-mail.
- Examine RAMP's risk information templates that document lesson-learned summaries and mitigation strategies on similar projects.
- Create custom portfolios of similar projects, their risks, and mitigation strategies.

One major drawback to RAMP is that this tool does not directly help software developers access their project risks in a quantifiable manner. There exist benefit in being able to consult with other practitioners and become aware of project risks that you may have been overlooked on your own software development project. Additionally, by examining patterns among the software risk, insights can be derived to help verify which software metrics influence software estimation.

5. Commercial Parametric Models

Parametric models generate estimates based on statistical relationships. Parametric models relate dependent variables (i.e., cost and/or schedule) to one or more independent variables (i.e., parameters). Parametric estimating techniques for software projects generally estimate overall system costs based on a software program's design characteristics. These overall costs can be partitioned among the life cycle phases. The advantages of parametric models are: they are fast and easy to use, they require little detailed information (after they are calibrated), and they capture total system or Computer Software Configuration Item (CSCI) level costs (including costs for integration activities). In addition, parametric estimating techniques can be as accurate (if not more accurate) as other estimating techniques, if they are properly calibrated and validated. Because of these advantages, parametric models are generally DOD's software estimating technique of choice. The discussions in the next section focus on several commercial parametric models that are widely used by both Industry and Government. [PEH99]

The [PEH99] provides a concise listing of three families of parametric commercial software models. The following is an extract of their handbook. There are many sophisticated parametric software estimating models that use multiple parameters to compute software costs and effort. This section discusses three common software parametric models and provides a basic understanding of some of their common attributes. The three models are the Constructive Cost Model (COCOMO), PRICE Software Model (PRICE S®, and Galorath Software Evaluation and Estimation of Resources Software Estimating Model (SEER-SEM®). For each of these models the background, principal inputs (i.e., parameters), and principal outputs are discussed.

a. COCOMO

Dr. Barry Boehm, formerly of the TRW Corporation, developed the COCOMO parametric software model and published its first edition in 1981. COCOMO is not a proprietary model and is completely described in Dr. Boehm's 1981 book [Boeh81]. In actuality, only a pocket calculator with an exponential key is required to execute COCOMO, although many computerized versions are available in the marketplace. During the last several years, Dr. Boehm made substantial revisions to COCOMO.

(1) COCOMO 81. COCOMO 81 (i.e., the first version of COCOMO) is a regression-based model that considers 63 programs in three modes: embedded, semi-detached, and organic (modes relate to the environment within which software development activities occur (e.g., experience related to systems; need for software conformance with established requirements; concurrent development with associated new hardware products)). Separate equations relating Level-of-Effort (LOE) in man-months (MM) to program size in thousands of delivered source instructions (KDSI) are established for each mode.

There are three levels within COCOMO 81: Basic, Intermediate, and Detailed. The Basic level consists of three simple models with single parameter effort and schedule equations. Effort equations relate MM to KDSI; and the schedule equations relate months of time needed (M) to MM, computed from the effort equations. The Basic level of COCOMO is often used to develop rough order of magnitude (ROM) estimates for software costs. Figure 7 lists effort and schedule equations for the three modes of basic COCOMO 81. The Intermediate COCOMO 81 contains nominal equations, also shown in Figure 7 which are similar to the basic equations for each mode (except for different effort coefficients), except they contain 15 multipliers, which adjust the result of the nominal equations to reflect a program's unique attributes.

Finally, the Detailed COCOMO 81 adjusts the multipliers for each phase of the software life cycle. All COCOMO levels are designed for use on any program, although the model is probably best used for CSCI-level estimates. All levels consider the CSCI phases previously described (e.g., software development cycle from

design through CSCI testing, although the design phase is partitioned into the "product design" and "detailed design" phases). Also, all three COCOMO 81 levels allow an adjustment to KDSI for reused or modified software. COCOMO 81 computes "effective KDSI" based on the percentages of redesign, recoding, and retesting required.

Mode	**Basic** ***Effort***	**Basic and Nominal** ***Intermediate Schedule***	**Nominal** ***Intermediate Effort***
ORGANIC	$MM = 2.4 (KDSI)^{1.05}$	$M = 2.5 (MM)^{0.38}$	$MM = 3.2 (KDSI)^{1.05}$
Semi-Detached	$MM = 3.0 (KDSI)^{1.12}$	$M = 2.5 (MM)^{0.35}$	$MM = 3.0 (KDSI)^{1.12}$
Embedded	$MM = 3.6 (KDSI)^{1.20}$	$M = 2.5 (MM)^{0.32}$	$MM = 2.8 (KDSI)^{1.20}$

Figure 7 Elementary COCOMO 81 Equations.

(2) COCOMO 81 Inputs. The following discussion focuses on the Intermediate level of COCOMO 81. The primary Intermediate COCOMO input (i.e., cost driver) is program size, in KDSI. However, there are 15 additional attributes that must be assessed. These attributes, shown in Figure 8, are classified into the following four categories:

- *Product attributes*: Product attributes describe the environment the program operates in. The three attributes in this category are: Reliability Requirements (RELY), Database Size (DATA), and Product Complexity (CPLX).
- *Computer attributes*: Computer attributes describe the relationship between a program and its host or developmental computer. The four attributes in this category are: Execution Time Constraints (TIME), Main Storage Constraints (STOR), Virtual Machine Volatility (VIRT), and Computer Turnaround Time (TURN).
- *Personnel attributes*: Personnel attributes describe the capability and experience of personnel assigned to the program. The five attributes in this category include: Analyst Capability (ACAP), Applications Experience (AEXP), Programmer Capability (PCAP), Programming Language Experience (LEXP), and Virtual Machine Experience (VEXP).
- *Project attributes*: Project attributes describe selected project management facets of a program. The three attributes in this category include: Use of

Modern Programming Practices (MODP), Use of Software Tools (TOOL), and Required Development Schedule (SCED).

	Rating					
Attributes	**VL**	**LO**	**NM**	**HI**	**VH**	**XH**
Required Reliability (RELY)	0.75	0.88	1.00	1.15	1.40	
Database Size (DATA)		0.94	1.00	1.08	1.16	
Product Complexity (CPLX)	0.70	0.85	1.00	1.15	1.30	1.65
Execution Time Constraints (TIME)			1.00	1.11	1.30	1.66
Main Storage Constraint (STOR)			1.00	1.06	1.21	1.56
Virtual Machine Volatility (VIRT)		0.87	1.00	1.15	1.30	
Computer Turnaround Time (TURN)		0.87	1.00	1.07	1.15	
Analyst Capability (ACAP)	1.46	1.19	1.00	0.86	0.71	
Applications Experience (AEXP)	1.29	1.13	1.00	0.91	0.82	
Programmer Capability (PCAP)	1.42	1.17	1.00	0.86	0.70	
Virtual Machine Experience (VEXP)	1.21	1.10	1.00	0.90		
Programming Language Experience (LEXP)	1.14	1.07	1.00	0.95		
Use of Modern Programming Practices (MODP)	1.24	1.10	1.00	0.91	0.82	
Use of Software Tools (TOOL)	1.24	1.10	1.00	0.91	0.83	
Required Development Schedule (SCED)	1.23	1.08	1.00	1.04	1.10	

Figure 8 Intermediate COCOMO 81 Software Development Effort Multipliers.

Tables included in Chapter 8 of Dr. Boehm's book [Boeh81] describe ratings from "very low" to "extremely high," that must be assessed for each of the 15 attributes identified above. For example, the reliability factor (i.e., RELY) would be rated "very low" if an error in a specific software system caused only slight inconvenience; "nominal" if an error could result in moderate, recoverable losses; or "very high" if potential loss to human life is at stake. Figure 8, extracted from Dr. Boehm's book, shows the numerical values assigned to specific ratings of Very Low (VL), Low (LO), Nominal (NM), High (HI), Very High (VH), and Extra High (XH) for each of the 15 attributes. (Note that all "nominal" attributes would have no effect on the effort required). Dr. Boehm's book provides detailed guidance on these attributes.

(3) COCOMO 81 Outputs. The output of the Intermediate COCOMO model is simply the LOE in MMs for the project being estimated, and a schedule in months. The effort output can easily be converted to a monetary value if the cost per MM is known. It is also possible to determine the allocation of the overall effort

to various phases of the CSCI life cycle, or time periods, using information presented in [Boeh81].

b. PRICE S®

The PRICE S® commercial model was developed by PRICE Systems, LLC to support software cost estimation. PRICE Systems, LLC also developed a hardware model, PRICE H®; a hardware operations and support cost estimating model, PRICE HL®; and a microcircuit and electronic module model, PRICE M®. The PRICE S® model is partly proprietary in that all equations are not published, though most are described in the PRICE S® Reference Manual. This model is applicable to all types of software projects, and it considers all software life cycle phases. In addition to the software life cycle phases, it also considers system concept and operational testing phases. Additional information on the PRICE family of tools can be obtained from the model vendor.

(1) PRICE S® Inputs. The principal inputs for PRICE S® are grouped into the following nine categories:

- *Project Magnitude:* Reflects the size of the software to be developed or supported. Size can be input as either SLOC, function points, or predictive object points.
- *Program Application (APPL):* Provides a measure of the type (or types) of software, described by one of seven categories: Mathematical, String Manipulation, Data Storage and Retrieval, On-Line, Real-Time, Interactive, or Operating System.
- *Productivity Factor (PROFAC):* PROFAC is a calibration parameter that relates the software program to the productivity and efficiency of personnel and management practices.
- *Design Inventory*: Provides for the amount of software inventory available for use (i.e., reuse). Two parameters: New Design (NEWD) and New Code (NEWC) indicate the amount of newness for each software type.
- *Utilization (UTIL):* Reflects the extent of processor loading relative to speed and memory capacity. Values above 50 percent usually increase effort.
- *Customer Specifications and Reliability Requirements:* The platform (PLTFM) parameter provides a measure of the level of testing and documentation that will be needed.

- *Development Environment*: Three complexity parameters (CPLX1, CPLX2, and CPLXM) measure unique project conditions such as multiple site development, requirements volatility, use of tools (e.g., CASE tools), and other factors.
- *Difficulty ratings for internal (INTEGI) and external (INTEGE) integration.*
- *Development Process*: Reflects the process being used. Choices include waterfall, spiral, evolutionary, and incremental development.

In addition, there are inputs specific to software support activities. These are discussed in further detail in the following paragraphs.

(2) PRICE S® Outputs. PRICE S® computes an effort estimate in person-months that may be converted to cost in dollars or other currency units. The effort is allocated among three stages of software development: Design, Code, and Test. The effort is also subdivided into five activities: Systems Engineering; Programming; Configuration and Quality Control; Documentation; and Program Management. PRICE S® also computes a development schedule in months, and provides a schedule effects option that compares an input schedule with that computed by the model. This option also shows penalties for compressing the user's schedule compared to the model's predicted schedule.

PRICE S® provides several optional outputs including resources-complexity and instructions-application sensitivity matrices, plus resource expenditure profiles. It also provides an "at-a-glance" output option to rapidly view the effects of changing selected input parameters. This option is useful for performing "what-if" type trade studies.

c. SEER-SEM®

SEER-SEM® is one of a family of tools offered by Galorath Associates. The family also includes hardware cost estimating and hardware-life cycle (SEER-HLC®) models, a software-sizing (SEER-SSM®) model, an integrated-circuit (SEER-IC®) model, and a design-for-manufacturability (SEER-DFM®) tool. SEER-SEM® is partly proprietary in that not all equations are published. However, some relationships are described in the SEER-SEM® User's Manual. SEER-SEM® is applicable to all program

types, as well as most phases of the software development life cycle. More information on the SEER family of tools can be obtained from the model vendor.

(1) SEER-SEM® Inputs. SEER-SEM® inputs can be divided into three categories: Size, Knowledge-Base Inputs, and Input Parameters. These inputs are described in further detail below.

- *Size:* Size can be input in one of three formats: SLOC, Function Points, or Proxies (proxies allow the user to specify his or her own size measure, which the model later coverts to SLOC). In addition, all software is categorized as "New," "Preexists Designed for Reuse," or "Preexists not Designed for Reuse." For pre-existing software, users must specify the amount of software deleted, plus the percentages of redesign, reimplementation, and retest required to modify or reuse the program for the current application. Because the model uses the Program Evaluation and Review Technique (PERT), users must input a "minimum," "most likely," and "maximum" value for all size inputs.
- *Knowledge-Base Inputs*: SEER-SEM® contains knowledge bases for different types of software. Knowledge bases assign default values to the input parameters described below, based on the type of software selected. Users must address these inputs to specify the knowledge base to be used by the model:
- *Platform:* The operating environment of the program (e.g., avionics, ground-based, or manned space).
- *Application*: The overall software function (e.g., command and control, mission planning, or testing).
- *Acquisition method*: The method in which the software is to be acquired (e.g., development, modification, or re-engineering).
- *Development method*: The method used for development (e.g., waterfall, evolutionary, or spiral).
- *Development Standard*: The standard used in development and the degree of tailoring (e.g., MIL-STD-498 weapons, ANSI J-STD 016 full, ANSI J-STD 016 nominal, or commercial).
- *Class:* (This input is primarily for user-defined knowledge bases.)
- *COTS component type*: The type of COTS program (if any), such as class library, database, or applications. (COTS relates to activities associated with incorporating commercial software components into development activities).
- *Input Parameters:* SEER-SEM® contains over 30 input parameters, from which users can refine their estimates. Similar to COCOMO 81 and

COCOMO II, the input values generally range from "very low" to "extra high." As in size, users must specify a "least," "greatest," and "most likely," value for each input. The selected knowledge base computes default values for most input parameters. Therefore, if users are unfamiliar with a particular parameter, they can use the knowledge-base default values. The primary categories of input parameters and a brief description of each follows:

- *Personnel capability and experience*: The 7 parameters in this category, similar to the "personnel attributes" of COCOMO 81, measure the caliber of personnel used on the project. These inputs are: Analyst Capability, Applications Experience, Programmer Capability, Language Experience, Host-Development System Experience, Target System Experience, and Practices and Methods Experience.
- *Development support environment*: The 9 parameters in this category are similar to the project attributes and some of the computer attributes in COCOMO 81. They include: usage of modern development practices, usage of automated tools, turnaround time, terminal response time, multiple site development, resource dedication, resource and support location, host system volatility, and target system volatility.
- *Product development requirements*: The 5 parameters in this category include: requirements volatility, rehosting from development to target computer, specification level, test level, and quality assurance level. The last three parameters are identical to the reliability attributes described above for COCOMO 81.
- *Reusability requirements*: The 2 parameters in this category measure the degree of reuse needed for future programs and the percentage of software affected by reusability requirements.
- *Development environment complexity*: The 4 parameters in this category measure the language complexity, host development system complexity, application class complexity, and the impact of process improvements.
- *Target environment*: The 7 parameters in this category are similar to some of the computer attributes of COCOMO 81, but focus on the target computer. These include special display requirements, memory constraints, time constraints, real-time code, target-system complexity, target-system volatility, and security (this is the most sensitive input parameter in the model).
- *Other input parameters*: There are also special inputs for schedule constraints, labor rates, integration requirements, personnel costs, metrics, and software support.

(2) SEER-SEM® Outputs. SEER-SEM® allows the users to select a variety of outputs. The model provides labor estimates in the categories of

management, systems engineering, design, code, data, test, configuration management, and quality assurance. A "quick estimate" provides a snapshot of size, effort, schedule, ETR, and other selected outputs anytime during the estimating process. Optional outputs include a basic estimate, staffing by month, cost by month, cost by activity, person-months by activity, delivered defects, and a SEI maturity rating.

C. SOFTWARE METRICS

1. Software Metrics Roadmap

This research addresses issues and problems associated with how the field of software engineering currently goes about collecting metrics for software projects. The research identifies the shortcomings in traditional metrics approaches, which are often driven by regression-based models for cost estimation and defects prediction. The authors feel that the traditional approaches provide little support for managers wishing to use measurement to analyze and minimize risk.

The fundamental problems with regression models often lean to a misunderstanding about cause and effect [Fent00]. They open with a short example of using regression analysis in predicting road fatalities. Using regression analysis, a model will indicate that the number of fatalities on the highway is less during the months of January and February. Interpreted in naïve manner, this could lend managers into falsely believing that it is safer to drive on the highways during these months. However other factors, ones that can be identified only by causal relationships, will indicate that there are several other factors that cause fewer fatalities during the mentioned months. The real reason for fewer fatalities is that weather is more likely to be bad during the winter months and bad weather can cause treacherous road conditions. In these circumstances fewer people drive their cars and when they do they tend to drive more slowly.

The authors notion of a causal model is telling the story that is missing from the naïve approach – software practitioners can use it to help make intelligent decision for risk reduction, and identify factors that can be controlled or influenced. Causal modeling can provide an explanatory structure to explain events that can then be quantified.

Much of software metrics has been driven by the need for resource prediction models [Nogu00, Boeh81, Putn92]. Usually this work has involved regression-based models in the form of effort = f(size). Ignoring the fact that solution size cannot possibly cause effort to be expended, such models cannot be used effectively to provide decision support for risk assessment.

The authors offer up an extended definition of software metrics and divide the subject into two components:

- The component concerned with defining the actual measures (in other words the numerical metrics)
- The component concerned with how we collect, manage and use the measures

The rationale for almost all individual metrics has been motivated by one of two activities [Fent00]:

- The desire to assess or predict effort/cost of development processes
- The desire to assess or predict quality of software products

The authors conclude that the existing models are incapable of predicting defects accurately using size and complexity metrics alone. Furthermore, these models offer no coherent explanation of how defect introduction and detection variable affect defect counts. According to [Fent00] the reasons for this are as follows:

- Size based metrics, while correlated to gross number of defects, are inherently poor predictors of defects
- Static complexity metrics, such as cyclomatic complexity, are not significantly better as predictors (and in any case are very strongly correlated to size metrics)
- There is a fundamental problem in the way defect counts are used as a surrogate measure of quality. Specifically, counts of defects pre-release (the most common approach) are a very bad indicator of quality

The authors introduce causal models. The great challenge for the software metrics community is to produce models of the software development and testing process which take account of the crucial concepts missing from classical regression-based approaches [Fent00]. Specifically we need models that can handle:

- Diverse process and product variables
- Empirical evidence and expert judgment
- Genuine cause and effect relationships
- Uncertainty
- Incomplete information

The authors believe that Bayesian belief nets (BBNs) were by far the best solution for their problem and conclude that statistical-based approaches do not provide managers with decision support for risk assessment and hence do not satisfy on of the most important objectives of software metrics.

The proposed research will utilize causal analysis and a statistical-based approach to correlate the metrics as described during phase two of the research. Valuable insights can be derived from capitalizing on previous research efforts in the field of Bayesian belief nets.

2. Moynihan

Tony Moynihan conducted a survey of a homogenous group of project managers that revealed a surprising diversity of risk management concerns. He approach his research by surveying 14 experienced application systems developers, all located in Ireland. In the interviews with each of the 14, he used a technique of personal construct elicitation. It works as follows; a personal construct is a bipolar distinction, or scales, which a person uses when contrasting different people, objects, situations, and so on. For example, say a personal distinction between dogs is the likelihood of whether a dog will bite you or not. So, when comparing dogs, or when confronted by a particular dog, you would likely think in terms of "Will it or won't it bite me?" [Moyn97].

To survey the program managers' constructs, Monahan asked each of them to list the development projects they had managed over the last couple of years. Then three projects from the list were selected randomly and the question was asked: In what important ways are any two of these three project the same, but different from the third, in terms of important situational factors you had to think about when planning the

project? This process was repeated with different triads of projects until all of the projects were depleted and no new constructs were introduced.

The personal constructs identified by the 14 Project Managers were then assigned to various themes and compared to Barki's [Bark93] risk variables and the Software Engineering Institute Taxonomy-Based Risk Identification instrument [Carr93]. The research continues to detail the differences, similarities, omissions, and one-to-one correlations between the constructs, Barki's, and SEI.

As one might expect, the program managers' personal constructs reflect most of the risk factors identified in the software project management literature. But their constructs also contain several rich elaborations to some of these factors, for example, the concept of requirements uncertainty. This is addressed by seven different personal constructs, each of which captures a subtly different but important facet of the concept. Given this level of elaboration on a single concept, we can only wonder if it will ever be possible to capture all the subtleties of projects on any reasonably sized checklist [Moyn97].

There exist other constructs that receive little attention in the mainstream literature. For example, take aspects of where the project's control resides and how it is exercised and aspects of the interface between developer and client organizations. These omissions may be a direct consequence of differences between the contexts in which the different studies upon which the literature is based were carried out. The notion of building a single, all-encompassing risk taxonomy for use by all software developers is probably unrealistic. Researchers need different risk taxonomies for different project contexts [Moyn97].

The author feels that if a larger body of project managers (beyond the original 14) would have been interviewed, it is unlikely that the broad findings would have been different. Even within the 14 interviews, the variation across program managers in the foci of their constructs provokes questions that can be only answered by further study:

- How, if at all, does a program managers' professional training or experience influence the way in which he or she construes projects?

- Do different program managers in the same organization tend, over time, to see things in much the same way and thus converge on the same constructs?
- Is it possible to build a useful taxonomy of program managers based on the foci of their constructs, one that yields categories such as the politician, the technician, and so on?

Better understanding the conceptual lenses with which project managers approach software projects – and the biases that tint those lenses – may help researchers isolate and avoid behavior that reduces management effectiveness which promoting behavior that increases it [Moyn97].

This research provides some interesting insights to developing or verifying a set of metrics in which to develop a risk assessment model. Additionally, the research suggests which measurements, currently found in existing literature, produce negligible effects on software risk. The proposed research pertains to software metrics and must consider the works of Moynihan as well as the results of the SEI and Barki. The research will be experimenting with a body of metrics available from the QSM database. Any insights that derived from these sources will certainly prove beneficial.

3. Dupont Scale

A Software Engineering Masters Student, [Dupo02], agreed to conduct research to provide a tailored complexity measure for PSDL. [Dupo02] documents his hybrid measure; specifically adapted for PSDL. This complexity measure provides a suitable interface to future development of the risk assessment model. The following extract is from [Dupo02].

> Measurement is quantitative; it is about counting. What can be counted in software? Flow graphs are an important part of software engineering so we can count nodes, edges, and repetitions. In actual source code, we can count LOC, timing constraints, data types, etc. However, if we were to count LOC and define it as our complexity measure, we probably could not gather enough information to describe its understandability or readability but it may say something about its maintainability.
>
> Choosing the right measure means understanding the individual characteristics of a particular measure, the individual characteristics of the software and the goals of the measurement. LOC is only one measurement and is representative of sizing measures. If complexity is

> interpreted as maintainability then knowing the length of a program may be useful. …
>
> We find that in order to derive a measurement you have to understand the principles of measurement theory (i.e., the mapping of empirical relations to formal relations). You must understand the principles behind scale types (i.e., nominal, ordinal, interval, ratio and absolute); that there is some admissible transformation between the numbers in your scale. Finally, stated assumptions (i.e., axioms) are important when defining the measurement.
>
> Measuring gives a quantitative result to an empirical understanding. It is easy to see one item bigger than another but that requires direct observation. Placing a value to that observation allows others the understanding of its relational composition. In time, and when standards of measurement have been met, the understanding becomes commonplace because a scale is built.
>
> The basic rules of measurement apply also to software complexity. A scale and scale type need to be decided to provide meaning to the measurement. We can't say that 40 degrees is twice as warm as 20 degrees, but we can say that 40 feet is twice as long as 20 feet. The idea of temperature versus length is intuitively and fundamentally different. This was not understood so well before standard practices upon which to measure temperature and length were decided and units of measurement placed.
>
> In the beginning, there were multiple methods and units available to measure temperature and length. There was no standard. So, too, is the case with software complexity today. In time, there may be standards by which all software complexity is measured. In time, we will know just how complex a program measuring 50 is. But 50 what? The standard methods will help but they will not curb the different units of measurement. The standards can be chosen somewhat subjectively so long as the formal/quantitative relation continues to map to the empirical relation. These relations are the only manner in which to establish the standards. Axioms (i.e., empirical assumptions) must be explicitly stated to understand the mapping that occurs.

The Dupont Scale calculates the complexity of PSDL using a hybrid complexity measure that properly accounts for data flow and the properties associated with each operator and data stream. The theory is, as more data is generated and flows between operators, the complexity increases. Moreover, each property represents a different level of complexity. Operators and data streams become more complex as more properties are

associated with them. Minimizing data flow and associated properties, increases the understandability of the prototype; hence, reducing the complexity.

Derived was a ranking of the properties using a set of weights. Each operator and data stream is assessed a total weight based on the sum of its weighted properties. This weight is added to one, to represent each operator and data stream as something greater than itself. Borrowed, was the theory of information flow that takes the product of the total number of fan_in and fan_out data streams as a function of each module (i.e. operator), representing the total possible number of combinations of fan_in data streams to fan_out data streams for the module. This product is then multiplied by the weighted value of its functional operator, providing a complexity for that operator. Finally, the total system complexity is calculated as the sum of operator complexities. Stepwise, the complexity is calculated in this way:

- Determine the properties associated with each operator and data streams.
- Calculate the weight of each operator and add it to one.
- Calculate the weight of each data stream and add it to one.
- Pick a single operator and determine which data streams fan_in and fan_out.
- Sum the weighted values of all fan_in and fan_out data streams for that operator. Multiply the two quantities.
- Multiply this product by the weighted value of the associated operator to find the complexity of that operator.
- Continue this process with each operator and then sum the complexities of all the operators to give the system complexity.

D. SIMULATION TECHNIQUES

The proposed research will utilize two primary simulation techniques: VitéProject and Crystal Ball. VitéProject utilizes an additional application called Visio. This section details how the original software risk assessment model utilizes VitéProject. The last simulation technique, Crystal Ball works as an add-in to MS Excel and provide Monte Carlo randomness.

1. VitéProject 2.0

VitéProject is a commercial modeling and simulation tool based on the Virtual Design Tool (VDT). Contingency theory is the foundation of the VDT directed by Dr. Raymond Levitt at Stanford [Jin96]. The underlying model of VitéProject, based on VDT-2, has been validated on three levels at the Center for Integrated Facility Engineering at Stanford University (CIFE):

- micro-level analysis, using toy problems
- meso-level analysis, using toy problems and experiments
- macro-level analysis, by testing for authenticity, reproducibility, generalizability, and prospective

A full accounting of VitéProject validation strategy can be found in the dissertation of [Nogu00] and [Thom99]. The SOFTWARE RISK MODEL development uses the VitéProject 2.0 in an attempt to apply CPM/Pert modeling techniques to software engineering. [Nogu00] parameterized VitéProject to adequately simulate sixteen different software project developments. The previous research implemented 30 simulations for each scenario development.

VitéProject has several limitations hampering its potential. First, VitéProject limits a maximum of 100 simulated runs for a given scenario. Second, the resulting data is presented in summary format. The summary format provides very limited ability to conduct histograms or a sensitivity analysis.

To change software project scenarios, users had to make changes to each individual activity/actors in a given scenario, before a new scenario could be simulated. This process, even for the smallest organizational structures, becomes time consuming an inconvenient.

[Alex01], in a communication with VitéProject Incorporated, discovered that Vité no longer supports VitéProject 2.0. Their new product is SimVision 3.0. Although the analytical engine within VitéProject is the same as SimVision, SimVision provides a much needed user interface enhancement. However, our research benefits from the decomposed architecture of the original VitéProject 2.0.

2. Visio 5.0

VitéProject modeling and simulation software requires a graphical input of the organization's structure. Visio provides the graphical interface for VitéProject 2.0. Visio opens all VitéProject documents via the VitéProject stencil, which is akin to a template in Microsoft® Word. The initial organizational structure for a given scenario, as well as all changes to a scenario or an organization's parameters occurs within Visio.

Users can initiate a simulation from Visio, via a VitéProject add-in, or from VitéProject itself. If the scenario is executed from Visio, all data in the scenario drawing is first written to the VitéProject database file prior to the simulation run. If the simulation is run from Vité, the simulation relies solely on the data currently saved in the VitéProject database file.

Figure 9 presents the simulated organization and the simulated software process used in the development of the SOFTWARE RISK MODEL. The process presents only four cycles of evolution. Each cycle contains the activities represented in Evolutionary Software Development [Nogo00].

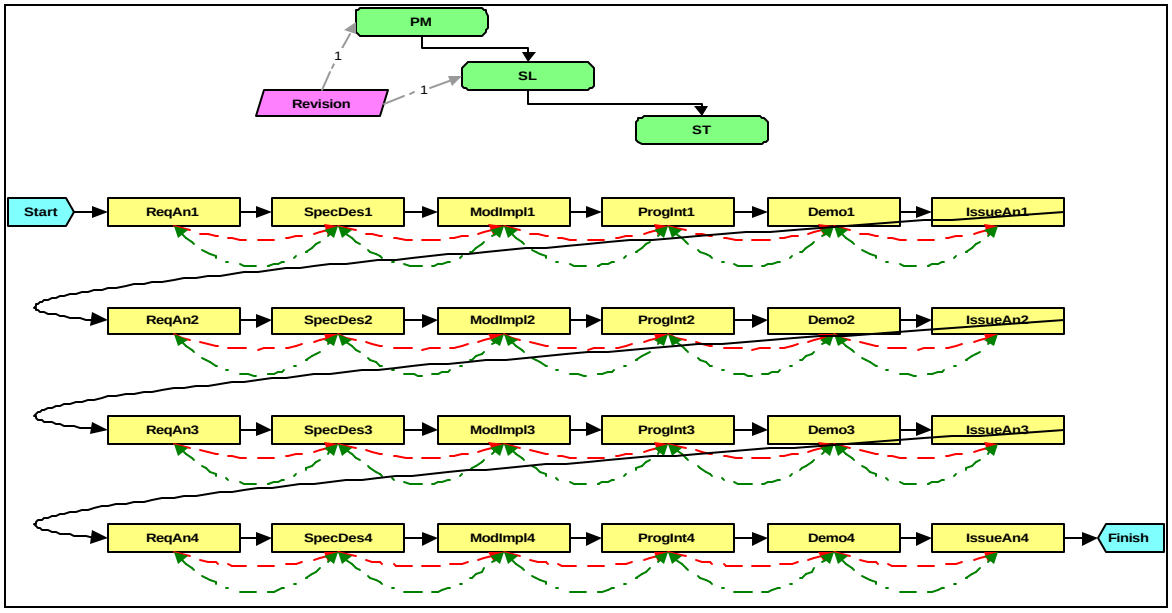


Figure 9 Organization Representation for VitéProject[2].

[2] The detailed description of the notation can be found on the VitéProject user manual [Levi99]. Rectangles indicate tasks. Rounded-corner rectangles indicate roles. Parallelograms indicate meetings. Double-headed-dashed arrows indicate information dependencies between tasks. Dashed arrows indicate problem dependencies between tasks. Normal arrows indicate precedence dependencies between tasks.

3. VitéProject Parameters [Nogu00]

The simulation data founding the SOFTWARE RISK MODEL was configured according to Figure 10 [Nogu00]. This research maintains the original integrity. However, the following extract from [Nogu00] needs additional explanations. This extracts provides the VitéProject parameter settings. Users should be able to duplicate the original work using Figure 10 and the probability settings provided in the extract from [Nogu00]). The remaining simulation parameters use the system provided default values.

VitéProject uses a set of default values for the variables of the model. These values are stored in a file named "behmatrx.opd" in the subdirectory of VitéProject. The behavior of the model depends on the values of these variables that are collectively called Behavior Matrix. This Appendix discusses the concepts considered in the behavior matrix and their relationship with software projects. The simulations used the default values for this file.

- Participant attention rule: Defines the probability distribution applied to the different selection methods (e.g. priority, FIFO, LIFO, random) of picking items to process.
- Participant tool selection rules: Defines the probability distribution applied to different information exchange tools (e.g. conversation, email, fax, memo, phone, video, voice-mail) given the type of message (e.g. Exception, Decision, etc.) A tool selected for an information exchange determines (1) the time needed for the message to move from one participant to another and (2) the time the message will stay in the in-tray of the receiver participant.
- Activity Verification Failure Probability (VFP) adjustment: There are two VFP (internal and external). The internal VFP depends on the complexity of the requirement and the skills of the participants. The external VFP depends on the complexity of the solution and the skills of the participants. The processing speed of responsible participants is affected by the solution complexity and the requirement complexity.
- Activity Information Exchange Frequency adjustment: This adjustment depends on the uncertainty of the activity and the team experience.
- Participant Processing Speed adjustment: This adjustment depends on the match between the participant and activity skill requirements.
- Definition of Rework, Quick-Fix, and Ignore decisions: This matrix defines how much of the original failed work should be reworked, quick-fixed or ignored. The values depend on the following failure types:

        - Internal|Internal: Amount of rework of an activity given internal activity failure (based on VFPInternal.).
        - Internal|External: Amount of rework of an activity given external failure (based on VFP External.).
        - External|External: Amount of rework of a failure dependent activity given external failure of an independent activity (based on VFP External of the independent activity.).
    - Impact of participant information exchange behavior on its VFP: This adjustment depends on the attendance or non-attendance of the participant to information exchange events related to the activities.
    - Impact of participant decision-making behavior on the VFP of failed activity: This adjustment depends on the centralization level of the organization.
    - The probabilities used by VitéProject were set as follows:
        - *Functional Error Rate* (0.01 low). Functional errors are the number of generated internal functional errors, shown in the Simulator Analysis Summary.
        - *Project Error Rate* (0.01 low). Project errors is the number of generated project errors, shown in the Simulator Analysis Summary.
        - *Information Exchange* (0.8 high)
        - *Noise* (0.1 normal)

Finally there is a set of matrices to implement Project Decision Making Policies including how to determine to whom to report an exception, how to make a decision for an exception, what is the maximum time a participant will wait before it takes delegation by default.

Figure 10 demonstrates how to represent the parameters required in the SOFTWARE RISK MODEL to the VitéProject. Consider for example *Requirements Volatility, RV*. The simulation setting for RV can be one of three values (Low, Medium, or High). If analysts intends on representing the RV as Medium, then set the VitéProject parameters of *uncertainty* and *requirement complexity* to Medium. This procedure is repeated for *efficiency* and *complexity.*

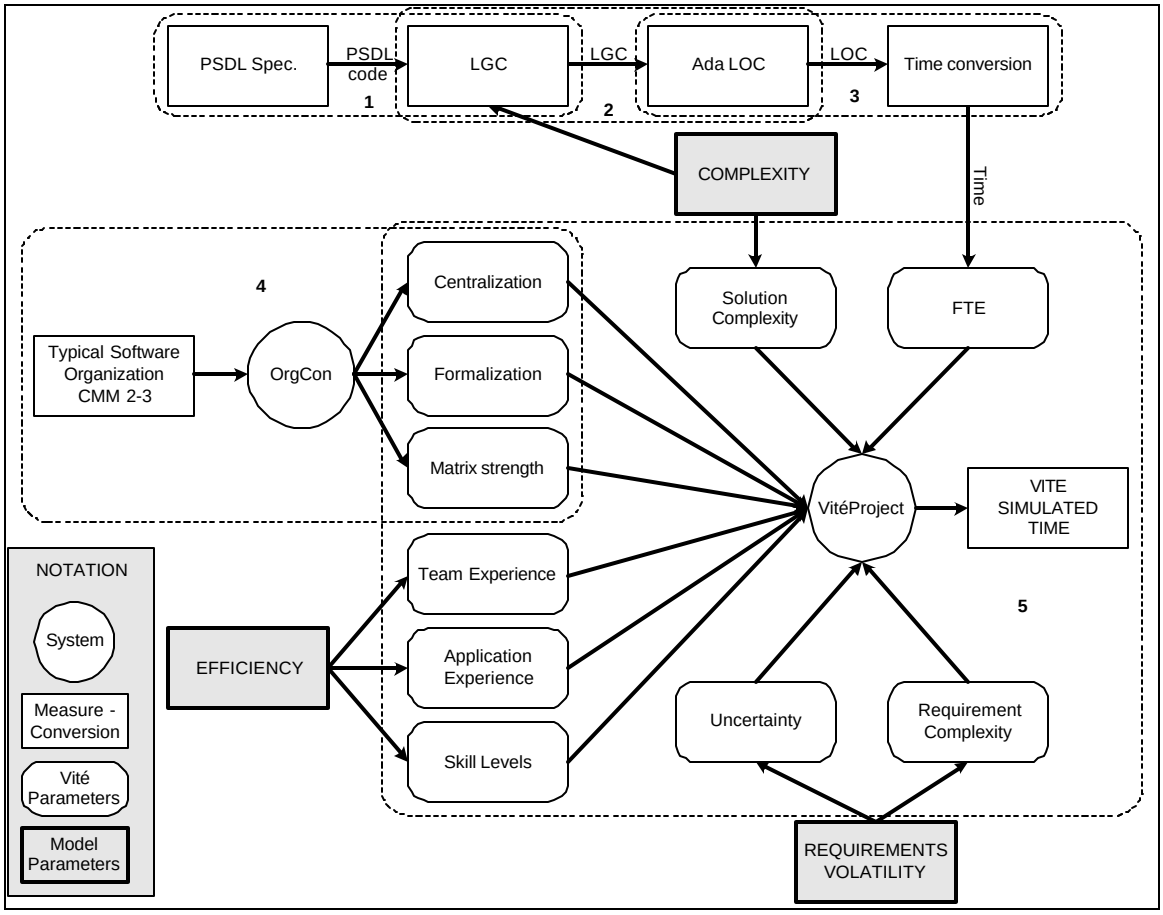


Figure 10 The Validation Project from "Nogu00".

4. Monte Carlo

The word simulation refers to any analytical method meant to imitate a real-life system, especially when other analyses are too mathematically complex or too difficult to reproduce. Without the aid of simulation, a spreadsheet model will only reveal a single outcome, generally the most likely or average scenario. Spreadsheet risk analysis uses both a spreadsheet model and simulation to automatically analyze the effect of varying inputs on outputs of the modeled system.

One type of spreadsheet simulation is Monte Carlo simulation, which randomly generates values for uncertain variables over and over to simulate a model. Monte Carlo simulation was named for Monte Carlo, Monaco, where the primary attractions are casinos containing games of chance. Games of chance such as roulette wheels, dice, and slot machines, exhibit random behavior.

The random behavior in games of chance is similar to how Monte Carlo simulation selects variable values at random to simulate a model. When you roll a die, you know that either a 1, 2, 3, 4, 5, or 6 will come up, but you don't know which for any particular roll. It's the same with the variables that have a known range of values but an uncertain value for any particular time or event (e.g. interest rates, staffing needs, stock prices, inventory, phone calls per minute).

For each uncertain variable (one that has a range of possible values), you define the possible values with a probability distribution. The type of distribution you select is based on the conditions surrounding that variable. The most common distribution types include *normal, triangular, log-normal, and uniform*; however several others exist. To add this sort of function to an MS Excel spreadsheet, you would need to know the equation that represents this distribution. A simulation calculates multiple scenarios of a model by repeatedly sampling values from the probability distributions for the uncertain variables and using those values for the cell. [Crys93]

THIS PAGE INTENTIONALLY LEFT BLANK

III. CONCEPTUAL FRAMEWORK

The conceptual framework of the proposed research involves promoting the confidence in the use of Nogueira's risk assessment model by validating the model against numerous post-mortem projects. A fundamental roadblock in this type of research is having accessible data which to conduct the research against. Considerable time and effort has been expended establishing a suitable mitigation to this issue.

Arrangements have been made to have access to the proprietary database and estimation tool suite SLIM developed and maintained by Quantitative Software Management (QSM®). QSM maintains a historical database of software projects development expanding the last twenty years. Currently, the database contains over 5,000 software projects. QSM is owned and operated by Larry Putnam. Mr. Putnam has extended an excellent research opportunity to utilize QSM's database while conducting the research.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESEARCH DESIGN

This chapter explains the details of the proposed research. Figure 11 below depicts the flow of research guiding the effort from theoretical foundations to the development of an enhanced model.

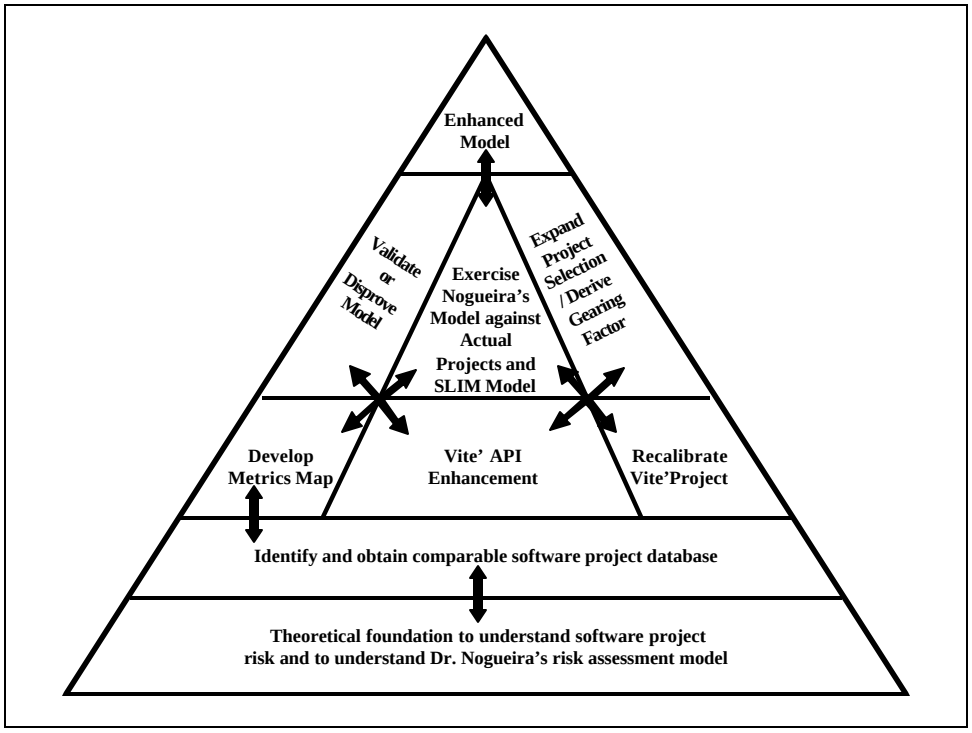


Figure 11 Pyramid of Research.

A. BACKGROUND

This research has begun by and will continue to evaluate the theoretical foundations of software metrics, software risk assessment, and software risk identification. Nogueira began this effort and committed an exhaustive effort when developing his model. His sources will be reviewed, any suitable omissions will be identified and researched, and research will close the gap between the conclusion of Nogueira's work and present day.

Additionally, it seems prudent to reconstruct Nogueira's findings. In order to conduct thorough research, complete understanding of the thought process and the decisions that went into the development of Nogueira's model must be demonstrated.

After this has been accomplished, enhancement and validation of the model can commence. The research must reexamine every facet of his work, to include the implementation details of the model itself. This research is being approached objectively with the intent to extend and validate the capabilities of Nogueira's risk assessment model.

B. PHASE 1 – PROJECT IDENTIFICATION

This phase involves determining the characteristics of the three projects that have been currently documented and identify comparable projects within the QSM database. This is a logical starting point to increase the chances of a successful mapping of the metrics. In essence, the documented research serves as a baseline. Since these are the only types of projects currently documented, this is where my research will begin.

The following is an extract from the Master's thesis [John01].

> After studying the Nogueira Dissertation, and before considering any candidate projects for this Thesis, we consulted with Dr. Juan Carlos Nogueira. As a result, we established a set of desired criteria to help in determining which were the most viable projects for application of the model.
>
> - The projects were recent DoD development projects utilizing the Software Engineering Institute, Capability Maturity Model, (CMM) level 2 processes or better which had the necessary metrics and data available.
> - The projects had worked through the lifecycle phases of an evolutionary development, in this case an evolutionary spiral model.
> - The project used Object-Oriented Methodology (OOM)
> - The project contained multiple Computer Software Configuration Items (CSCIs)
> - The project was Coded in Ada
> - The software was real-time embedded
> - The project used a Computer Aided Software Engineering, (CASE) tool.
>
> Two projects were used for the research:
>
> The first project used an Evolutionary Spiral lifecycle model. It used Object-Oriented methodology. It was composed of five Computer Software Configuration Items. It used Ada. It was real-time embedded, it used Rational Rose as a Computer Aided Software Engineering tool, and the developer was operating at SEI level 3. In addition, software metrics

from three builds over a period of three years had been kept. This project met all of our ideal project selection criteria; we selected this as our first candidate project.

The second project originally used an incremental build lifecycle model and not an evolutionary model. It originally used Functional Decomposition methodology. It was composed of six Computer Software Configuration Items. It used Ada and assembler. It was real-time embedded. It did use upper CASE tools, like Requirements Traceability Matrix (RTM), it did not use lower CASE tools such as Rational Rose. The development effort initially was performed in an Ad-hoc manner with little software process involved and had experienced extreme volatility and poor metrics early in its development. However, due to a major restructure and overhaul of the project, and a shift of focus to institutionalizing software processes, (SEI CMM level 3 certification), the project migrated to Ada, and began using a modified Incremental Build lifecycle model. In addition, suitable software metrics from two recent builds were available. Although this project did not meet all of our ideal project selection criteria, i.e., it did not use a lower CASE tool and had limited available metrics, we selected it as our second candidate project because we felt the data was sufficient to analyze and draw conclusions.

The following extract is from the dissertation of Nogueira [Nogu00].

The model was applied to a large project developed by the Uruguayan Navy (the project is a simulator developed for war gaming (SIMTAS) consisting of 75,240 lines of code in Pascal)... … the model predicted 17 months instead of 18.

In an effort to establish the feasibility of finding suitable projects, I consulted with Larry Putnam of QSM and asked him to check the availability of projects meeting the following criteria, which are derived from the documented case studies above.

- Development organization at CMM Level 2 or above
- The projects were developed using an evolutionary development
- Object-Oriented Methodology preferred
- The project was coded in Ada
- Software was real-time embedded

A caveat to the request to QSM was the following, “If a project meets most prerequisites, but not all, this is also acceptable. Let's try to at least have Ada projects (Ada83 or Ada95) on real-time embedded projects.”

QSM responded indicating they should have better than a hundred Ada projects. The CMM level of the development organization would have to be inferred, QSM metrics do not directly capture this. Inference would also need to be used to determine the software development process. There are some projects that were developed with object-oriented technology, however; projects are also represented in the data that were developed prior to the dominance of object techniques.

The result of the initial request was a subset of 112 software projects. To provide the reader a better understanding of the quality and availability of data, some statistical information is provided. A more comprehensive statistical reference is provided in Appendix B.

Figure 12 details how the total number of projects are categorized according to the application type. The majority of the projects are from the Avionics industry while the least number of projects represent Microcode and Real-time systems.

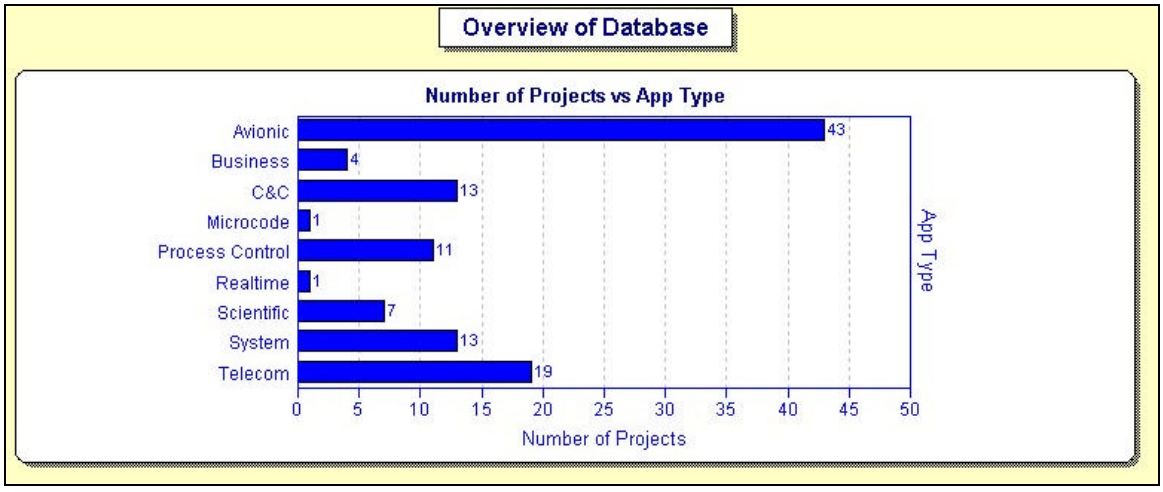


Figure 12 Projects vs. Application Type.

In an effort to preserve proprietary information, a generic identifier has replaced organization names. In the subset of initial projects available for evaluation, there are a total of 26 different organizations. Figure 13 displays the number of projects captured in the database by each organization.

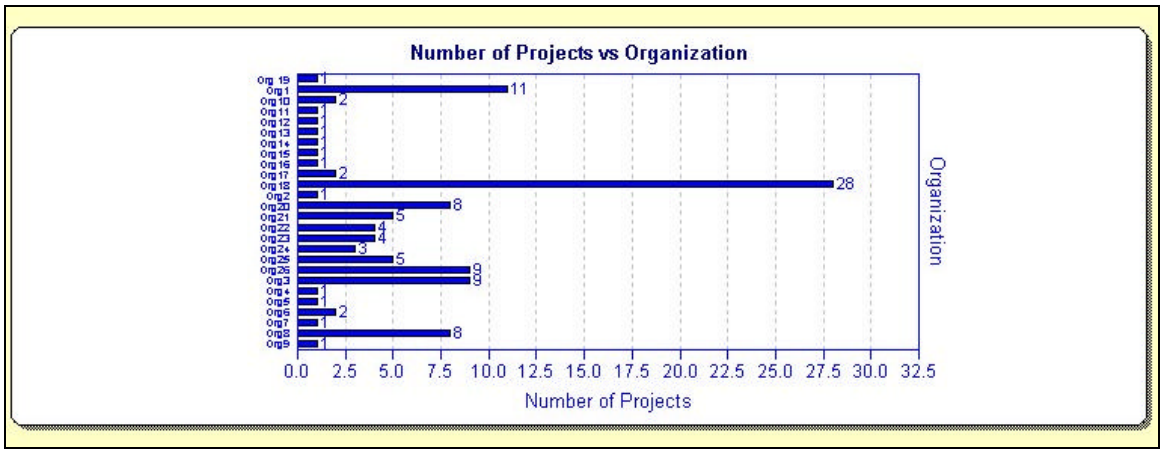


Figure 13 Projects per Organization.

To further clarify the effort expended to produce the software projects, Figure 14 considers the actual expended calendar time to complete each software development phase.

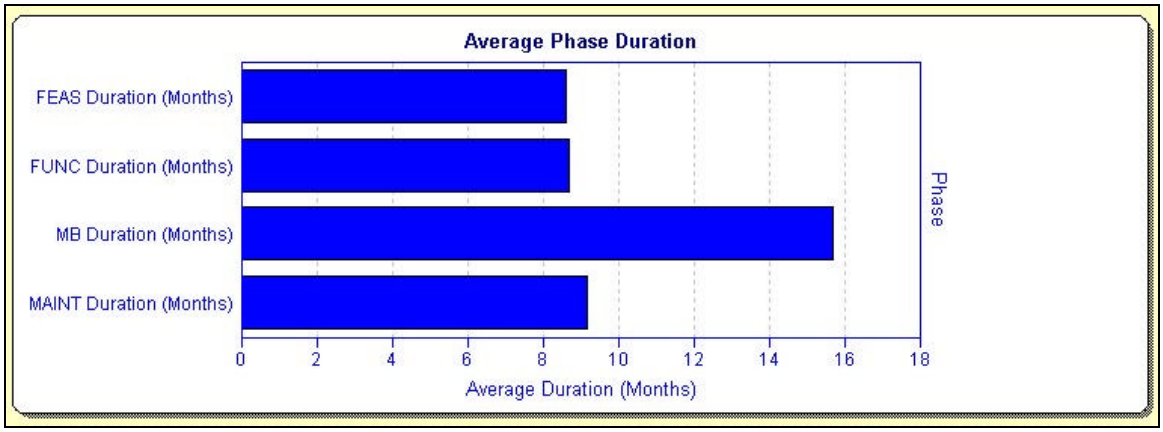


Figure 14 Average Phase Duration.

Finally, it is sensible to express how schedule slippages are captured in the database, see Figure 15. For example, during the main build, 33% of all of the projects ran over their initial estimation by as much as 25%. Probably more alarming is the fact that out of the 112 projects under consideration, only 13% finished on time or earlier than their schedule estimates.

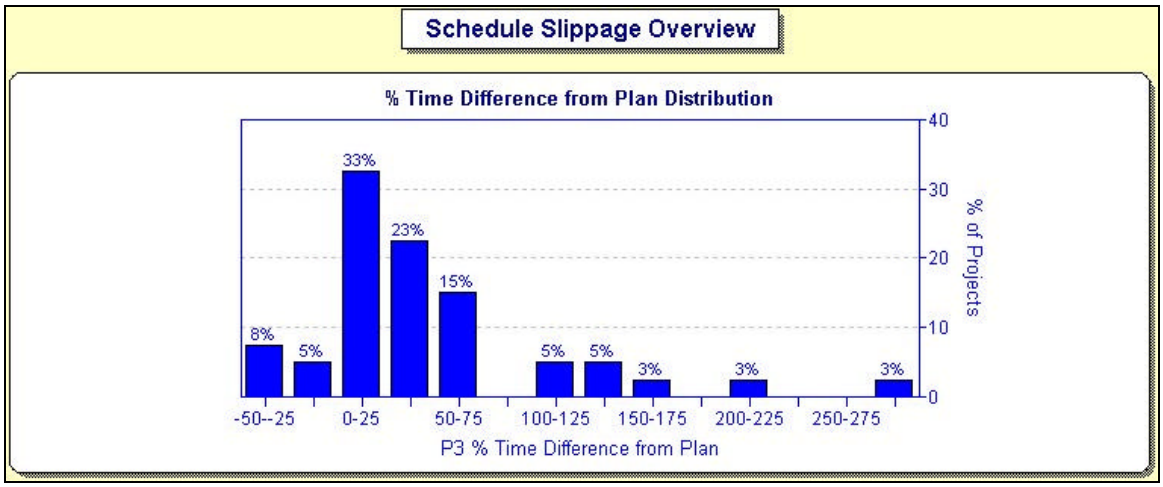


Figure 15 Schedule Slippage Overview.

C. PHASE 2 – DEVELOP THE METRIC MAP

In order to proceed with the research, it will be necessary to determine correlations between the available software project data and the metrics required for Nogueira's risk assessment model. Nogueira utilizes three primary metrics: requirements volatility, efficiency, and complexity. Metrics, in this form, are not available for the research in the QSM database. A complete listing of the "default" metrics available in the QSM database is included in Appendix A.

Finding correlations among the metrics will be the most involved and consume the majority of the research effort. Presently, I am not exactly sure how to approach this issue. Initial thoughts are to consider techniques using Bayesian belief nets, factor analysis, sensitivity analysis, and/or generalization per category. These potential threads are not mutually exclusive. The problem with developing a mapping between the available data and the required data in Nogueira's model is divergence will be extremely difficult to isolate.

Statistical tools are also available for the research from the QSM SLIM suite of products. These tools are currently used to analyze the QSM database. Other possible statistical tools could come from SAS Institute Inc. and SPSS Inc., but a thorough evaluation of these tools to establish their applicability and usefulness for the research has not been conducted to date.

D. PHASE 3 – EXERCISING THE MODELS

A problem in planning the research is determining when to proceed to phase three, exercising the models. In actuality, Phases two, three, and three (a) will initially be worked in conjunction with each other. Nogueira documented that his model produces valid results in the following ranges of accuracy [Nogu00].

Maximum error = 127 days (28%)
Average error = 19 days
Error standard deviation – 23 days
5% of projects with error >= 25%
65% with 5% < error < 25%
30% with error < 5%

The initial idea is to apply causal analysis to the available metrics in the database and reproduce results within these ranges. Three basic artifacts are available to work with, all of which can provide software estimations. The first artifact is Nogueira's risk assessment model. The second artifact is the collection of projects captured in the QSM database. Third, the estimation tools available in the QSM SLIM suite of tools. Using these three artifacts, estimation results can be derived and documented in a redundant fashion.

E. PHASE 3A – METRIC MAP ITERATIONS

As mentioned in phase three above, this phase is an iterative process of exercising various combinations of the available metrics to find suitable correlations to the required metrics for Nogueira's model. Initially causal analysis will be applied, and then perhaps, experiments will lend themselves to a more scientific method to map the metrics.

F. PHASE 3B – RECALIBRATION OF VITE'PROJECT

The first approach to establishing a correlation in the metrics will involve experiments with cause and effect combinations. However, this may not prove to be sufficient. In phase three (b), the original scenarios used in the Vite'Project parameters will be reestablished using actual project data instead of the configuration utilized by Nogueira. The bearing of this recalibration remains to be derived, but it's reasonable to consider this could impact the margin of error represented in the model.

Consideration was given to conducting this phase of the research prior to beginning the mapping of the metrics (phase two). An assumption going into this dissertation is that Nogueira's risk assessment model is valid; the model just needs to be proven. With this in mind, modifying Nogueira's model as a legitimate first step is not warranted without reasonable cause.

G. PHASE 3C – VALIDATION OF THE NOGUEIRA MODEL

After exhausting phases three (a & b), research may lead to examination of Nogueira's model with closer scrutiny. If deviation continues to present itself when conducting phase three, I may have essentially resort to "ground zero" to establish potential conflicts.

It should be noted that phases three (a, b, & c) should not be considered mutually exclusive. Research could indicate that partial modifications are required in all three sub-phases. Additionally, all of the sub-phases conducted during phase three are subject to the impact of the introduction of the Vite'Project API. This automated tool will improve the statistical significance obtained when utilizing the Vite'Project simulation, greatly increasing the number of simulation runs provided by the simulation.

H. PHASE 3D – DEVELOP GEARING FACTOR

The proposed research will continue into phase three (d) after Nogueira's risk assessment model has been successfully exercised using the project data of the 112 projects identified in phase one. Either a successful metric map, a reprogramming of the Vite'Project simulation, an enhancement to Nogueira's risk assessment model, or a combination of any and all of these factors, has accomplished confidence in the model. Only at this point will I extend the project base to include additional projects to further reduce the margin of error produced in the model.

As the reader will recall, the QSM database has accumulated in excess of 20 years and contains upwards of 5,000 projects. During this phase of the research I will continue to exercise the model against these additional projects. Depending on the implementation details required to exercise the additional projects, database exhaustion may be feasible. Initially, categorizing the projects according to their industry sector and their

development language seems logical. There may exist further merit in classifying the software projects according to their software development lifecycle.

Experiments may indicate that Nogueira's model is sufficient enough to support this wider range of software projects in its present form. The hypothesis is that it may be necessary to develop a factor that is used to adjust the results of Nogueira's model when it is applied to other domains. This factor is being referred to as a gearing factor. This definition of a gearing factor is different that the definition utilized in QSM database (Appendix A).

I. PHASE 4 – ENHANCEMENT OF NOGUEIRA'S MODEL

Phase 4 is a culmination of all of the results derived from the previous phases. The intent is to deliver an enhanced model whose performance exceeds the acceptable ranges explained in the introduction of phase three. The end state of the proposed research is an enhanced version of the Nogueira model, validated not only in the original software project domains, but extended to produce acceptable results when applied to any software development activity.

THIS PAGE INTENTIONALLY LEFT BLANK

V. PRELIMINARY RESEARCH

Prior to exercising the software risk model with real project data, a series of experiments were conducted using a Monte Carlo simulation. The purpose of the experiments are to determine the sensitivity of the software risk model's input parameters in addition to determining the model's behavior and parametric limitations.

Additionally this chapter presents the initial analysis of using real project data. In essence, the real projects were used to derived frequency distributions. This frequency distribution provide input assumptions to the Monte Carlo simulation. Additionally, frequency distributions are conducted on the project durations of the actual data. The intent is to demonstrate the level of error between the software risk model's projections and the actual project data.

A. SOFTWARE RISK MODEL BEHAVIOR

To apply a Monte Carlo simulation to the software risk model, the model assumptions (i.e. inputs) have to be examined to determine the most likely input distribution for each. The software risk model can be considered to have four assumptions: Efficiency, Requirements Volatility (Birth Rate and Death Rate), and Large Granular Complexity. The forecast from the Monte Carlo simulation will monitor the required number of calendar days projected from the software risk model[3].

1. Efficiency

Efficiency is determined by calculating the ratio of *direct time* to *idle time.* Essentially if an organization is exhibits a 70% direct time, leaving a 30% idle time, then the derived Efficiency becomes 2.33. [Nogo00] documents that an organization can be considered a "high" efficiency organization if the derived Efficiency ration is greater than 2.0. In the proceeding example, this organization is considered "high" efficiency.

Figure 16 below documents the distribution exercised to simulate the Efficiency of an organization. In this case, a triangle distribution is utilized to simulate the direct

[3] The software risk model can calculate calendar day projections when supplied a confidence interval in-lieu-of the number of calendar days. Our research utilizes a confidence interval of 95%.

time (the numerator) of the Efficiency equation. The denominator, idle time, is calculated by $IT = (100 - DT)$. We assume that the majority of organizations operate with approximately 85% direct time and no organizations operate below a 40% direct time.

Assumption: Direct Time **Cell: E8**

Triangular distribution with parameters:

Minimum	40.00
Likeliest	85.00
Maximum	100.00

Selected range is from 40.00 to 100.00
Mean value in simulation was 74.99

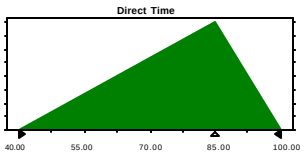


Figure 16 Direct Time Distribution.

2. Requirements Increase

The increase of requirements help derive the overall Requirements Volatility of the software project. This measure cannot be a negative number but can exceed upwards of the 100 to 200 percent. Figure 17 details how the requirements increase in projected. We assume the majority of the software projects enjoy a modest increase of 20% requirements increase.

Assumption: Requirements Increase **Cell: E9**

Lognormal distribution with parameters:

Mean	20.00
Standard Dev.	20.00

Selected range is from 0.00 to +Infinity
Mean value in simulation was 20.32

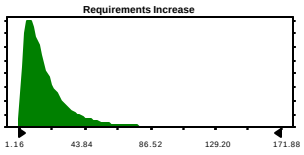


Figure 17 Requirements Increase Distribution.

3. Requirements Decrease

The decrease of requirements also helps derive the overall Requirements Volatility of the software project. Like requirements increase, this measure cannot be a negative number but is not expected to reach the levels of the increase requirements

(except in project cancellations or major restructuring). Figure 18 details how the requirements decrease is projected. We assume the majority of the software projects enjoy a modest increase of 5% requirements decrease.

Assumption: Requirements Decrease **Cell: E10**

Lognormal distribution with parameters:

Mean	10.00
Standard Dev.	9.00

Selected range is from 0.00 to +Infinity
Mean value in simulation was 9.94

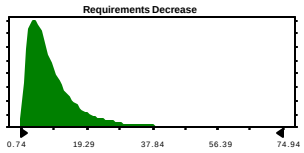


Figure 18 Requirements Decrease Distribution.

4. Large Granular Complexity

The Large Granular Complexity is a measure of the overall relational size of the software project [Dupo02]. For our simulations we are assuming a level distribution for the software project size in the ranges of 5K to 650K E-SLOC. Figure 19 details the uniform distribution and identifies that our simulation trials experience a mean of 326 thousand effective source lines of code.

Assumption: E-SLOC **Cell: E12**

Uniform distribution with parameters:

Minimum	5.00
Maximum	650.00

Mean value in simulation was 326.16

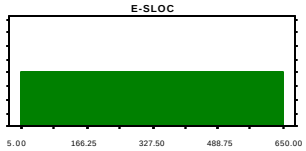


Figure 19 E-SLOC Distribution.

5. Generic Simulation Performance

With the establishment of the assumption distributions the analysis can continue to exercise the software risk model. For our analysis, we conducted 10,000 unique simulations. These simulations were conducted with each of these four assumption distributions. The resulting forecast are presented in Figure 20 below.

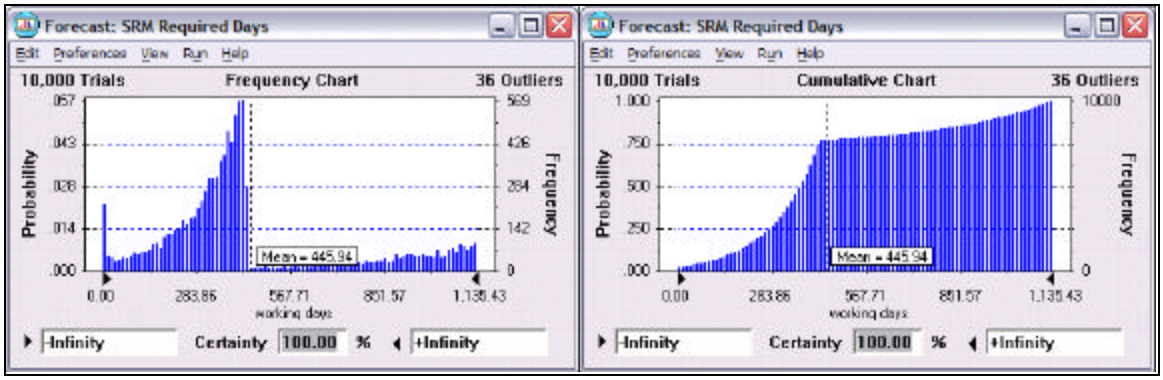


Figure 20 Software Risk Model Performance on Simulated Assumptions.

The distribution on the left-side of Figure 20 illustrates the frequency chart of the 10K simulations. The right-side illustration is the cumulative distribution (cdf). Figure 20 illustrates some intriguing behavior of the software risk model. Specifically, with these assumptions (the distribution limits), there exist a strong probability of projecting a requirement of zero days to complete the project.

Probably more interestingly is the steep reduction in frequency around 440 days. This apparent discontinuity requires additional investigation. Figure 21 below illustrates that approximately 77% percent of all of the projects were projected to be complete by the mean amount of days (446). Essentially this indicates that only 23% of the projects will ever extend past 20 months[4].

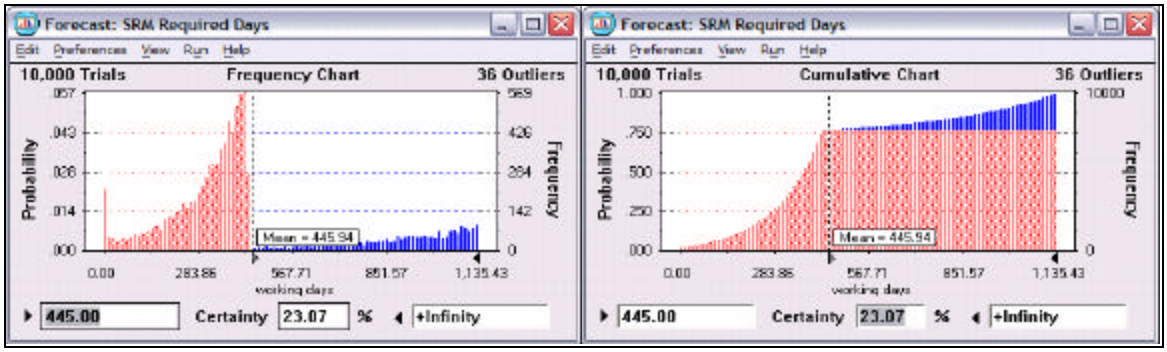


Figure 21 Software Risk Model Performance on the Mean.

The distribution of the 10K projects indicates the possibility of two distinct situations. First, there exist subsets of input assumptions that cause the software risk

[4] [Nogu00] indicates that a calendar month is equivalent to 22 working days.

model to project in the range of zero days to approximately 440. Second, an additional subset of input assumptions causes the software risk model to extend the range of possibilities out through 1135 days (4.2 years).

The analysis on the software risk model behavior is curious in the least. We became interested to see how the model would perform against actual projects.

B. PROJECTIONS OF ACTUAL PROJECT DATA

Interesting results obtained during the preliminary research in Phase Two are documented here. This phase involved identifying suitable metrics to map from the QSM® data dictionary and applying these metrics to the original risk assessment model in order to document the model's accuracy. Appendix A contains QSM® data dictionary.

Projects in the QSM® repository populate an extensive amount of data fields. However, only a small fraction of the data base metrics appears amenable to analysis when implementing the risk assessment model. The following, Figure 22, is a partial list of available metrics from QSM®. The colored cells are the actual measures that were considered, in some fashion, to interface with the risk assessment model.

Project Name	Status	Confidence	Record Creation Date	Last Modified
Application Type	Description	Size	New	Modified
Unmodified	Requirements	Productivity Index	Defects First Month after Delivery	Phase Acronym
Start Date	End Date	Months	Effort	Cost
Peak Staff	Staffing Shape	Organization	Division	Country
Design Complexity	Development Classification	Industry Sector	Function Unit	Counting Method
Gearing Factor	Language	Percent of Total Size	Language Type	Effort Units
Person Hours per Month	Labor Rate	Labor Rate Unit	Development Environment	Operational Environment
Day / Week	Hours / Day	Defects	Time (months)	Effort
Cost	Peak staff	Size and Requirements Growth	System Benefit / Effectiveness	Significant Factors

Majority of Available Measures from QSM®

Figure 22 Available Measures from QSM®.

Due to limitations in the interoperability, there is not one unique mapping that can be established to exercise the risk assessment model. This caused me to research several potential metrics mapping scenarios. After studying the available metrics from QSM® and the original metric specification of the risk assessment model, I developed as set of 27 unique test cases. The intent is to map each of the 112 projects, according to each of the individual test case scenarios, and document the accuracy of the risk assessment model. The 27 unique test cases range from the most logical interpretation of the metric definitions to an exaggerated modification of the metric elements.

Figure 23 is a demonstration of how the available metrics from QSM® were mapped to the risk assessment model.

Legend	***EF***	***PI Breakpoint***		***CX***			
	A	*10*		*A*	*Conversion KLOC = 40 * LGC*		
	B	*13*		*B*	*Conversion KLOC = 30 * LGC*		
	C	*18*		*C*	*Conversion KLOC = 20 * LGC*		
	RV						
	A	*Req Growth % = Number used*					
	B	*0's in Req Growth replaced by 25%*					
	C	*All Req Growth % increased by 25%*					

Figure 23 Legend for Interface.

Recall that the risk assessment model requires three primary elements to derive a projection of software completion. The primary elements are Efficiency, Requirements Volatility, and a Complexity measure[5]. Additionally, either a desired number of days[6] or a desired completion percentage is supplied to the risk assessment model. The next section reviews the three primary risk assessment elements and describes how measures from the QSM® database are mapped.

[5] The first digit in the legend is *always* the efficiency and the second digit is *always* the requirements volatility. A third digit, when dealing with the SRM is the complexity.

[6] Do not confuse the number of days as effort in man-days. The SRM does not project the risk based on effort. Time in days is considered to be working days. The practice of converting working days to calendar months is 22 working days per month (Nogu00).

1. Efficiency

Efficiency (EF) – The following is extract from Nogueira's dissertation [Nogu00].

> The efficiency of the organization can be measured observing the fit between people and their roles in the software process. The skill match between the person and the job is required to estimate the speed in processing information and the rate of exceptions, which in turn affect the efficiency. The number of people and the turnover affect the efficiency as consequence of productivity losses due to training, learning curves and communications…

Low and high efficiency scenarios are determined by the ratio of the percentage of direct time over the percentage of idle time. The resulting ratio is calculated by

$$EF = Direct\%/Idle\%$$

The breakpoint for the low and high efficiency indicator occurs when idle time exceeds approximately 33%. Low efficiency organizations, idle time in excess of 33%, produce a resulting EF ratio less than 2.0. Calibrated ranges for a low efficiency organization are approximately 33% - 55% idle time. High efficiency organizations are organizations with idle time less than 33%.

The primary data field mapped from the QSM® database to represent EF is the productivity index. The following extract is from a QSM® training manual.

> Organizations represented in the database, use a metric that is called a "productivity index" or PI. The PI of an organization, for a particular project, is a management scale from one to 36, corresponding to the productivity parameter, that represents the overall process productivity achieved by an organization during the main build. The PI is a measure that quantifies the net effect of everything that makes projects different from one another.

In order to map the recorded productivity index to the risk assessment model, I implemented the logic that if an organization exceeded a particular productivity threshold, then the organization is considered either a high or low efficiency organization. The average productivity index in the data is around 10 ½. Using the average as a point of reference, I exercised the experiments with three different breakpoints.

As referenced in the legend, Figure 23 above an organization operating with a productivity index greater than 10, 13, or 18 would be considered a highly efficient

organization respectively. Therefore, the documented results demonstrate the sensitivity of the risk assessment model to adjusting these factors.

2. Requirements Volatility

The SRM utilizes a measure intended to capture the complications experienced during requirements elucidation. This measure is determined by recording the percentage of requirements from two different aspects. First it becomes necessary to determine the percentage of requirements that have been eliminated from a software project (Death Rate %). And second, determine the percentage of requirements that have been added to a software project (Birth Rate %). A combination of these two percentages yield a requirements volatility metric, as a percentage. This metric is illustrated below.

$$RV = BR\% + DR\%$$

In the database from QSM® there exist a measure that is collected called Requirements Growth Percentage. This measure documents how much the project requirements changed from the original plan.

The mapping between Requirements Growth Percentage and the Requirements Volatility occurs in three configurations. In the first configuration, I set up the experiments to reference a direct one-to-one mapping. The second configuration replaces any zero and non-entries in the QSM®® database with a baseline of 25, leaving the values greater than zero unaltered. The final configuration is a modification of the one-to-one mapping in which all Requirements Growth Percentage entries in the QSM®® database are increased by a static value of 25 for the Requirements Volatility.

3. Complexity

The final primary element that is required to utilize the risk assessment model is a measurement for the complexity of the software development. Nogueira derives a complexity measure from analyzing a Prototype System Description Language (PSDL) specification. This complexity measure is determined from the following equation.

$$LGC = O + D + T$$

The equation accounts for all of the operators (O), data streams (D), and abstract data types (T) contained within a flattened PSDL source code and combines their frequency to determine a measure called the Large Granular Complexity (LGC). The

LGC is predominately limited to analyzing PSDL code; however the development of the SRM provided a conversion equation for lines of Ada code. Nogueira derived the following conversion for Ada source code to LGC.

$$KLOC = (40LGC + 150)/1000$$

or roughly

$$KLOC = 40LGC/1000$$

The QSM® database provides excellent information detailing the coding effort of software projects. Information is available that documents the percentage of new, used, and modified code as well as the total counts in lines of code. The capability exists to segment the project data by development language. The mapping strategy involves calculating the total amount of Ada code (new or modified) applied to the software project and converting this value to LGC.

Preliminary results indicated that SRM conversion from KLOC to LGC could possibly be too conservative. Due to this fact, I set up additional scenarios to test the conversion formula for

$$KLOC = 30LGC/1000$$

and

$$KLOC = 20LGC/1000$$

The end result is three different configurations for each of the three primary elements required for the risk assessment model, for a total of $3^3 = 27$ possible combinations of mapping scenarios. In addition to the 27 combinations, the predominant coding language is Ada. The life cycle phases considered span the completion of the specifications thru the delivery of the code. We exclude the maintenance phase from the validation experiments.

4. Assumption Distributions

As with the first section of the chapter, we now have to determine the probability distributions to represent each of the required input assumptions. For the purposes of the

Monte Carlo simulation, we are only going to implement Scenario AAA[7] of the 27 mapped scenarios of Figure 23.

The approach is a little different when representing the actual project data. Instead of applying a generic distribution for each assumption, we are going to calculate each distribution from the QSM® database. We are using the data points from the initial 112 projects to ultimately determine the probabilities for 10K simulation trials.

Additionally, we are calculating the distribution of the required days in the project database. These will serve as a baseline to make our comparison against**.** Figure 24 illustrates the process. Each assumption has a corresponding probability distribution.

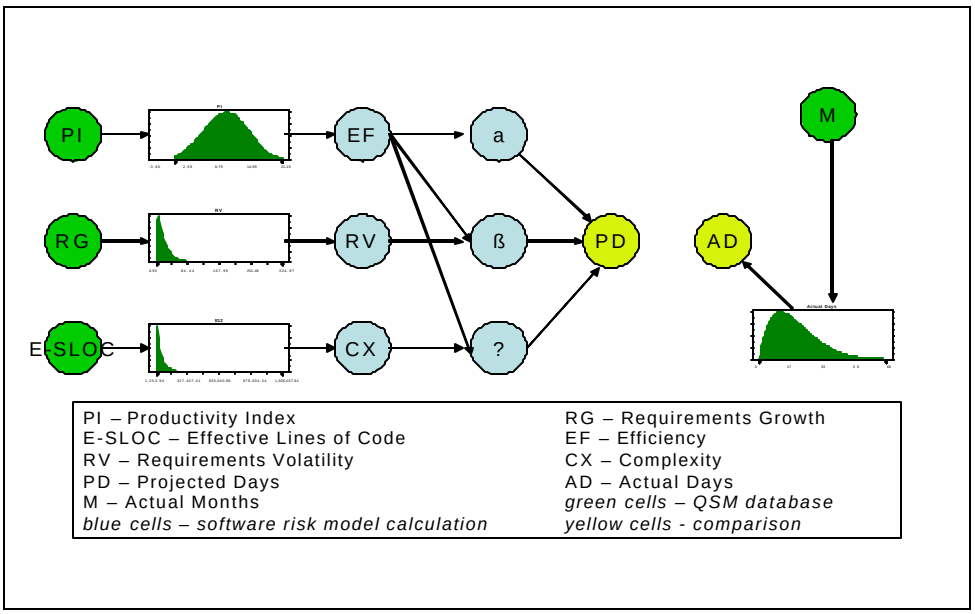


Figure 24 Simulation Architecture.

a. Productivity Distribution

Efficiency is determined by calculating the ratio of *direct time* to *idle time.* Essentially if an organization is exhibits a 70% direct time, leaving a 30% idle time, then the derived Efficiency becomes 2.33. [Nogo00] documents that an organization can be considered a "high" efficiency organization if the derived Efficiency ration is greater than 2.0. In the proceeding example, this organization is considered "high" efficiency.

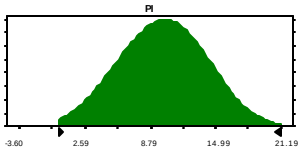
[7] This is the most logical scenario as documented by [Nogu00]. The Efficiency breakpoint with be a PI of 10. Requirements Volatility is utilized exactly as represented in the database. And the E-SLOC is converted over to LGC using a coefficient of 40.

Figure 25 below documents the distribution exercised to simulate the Efficiency of an organization. In this case, a triangle distribution is utilized to simulate the Productivity Index. As explained in the previous section, an organization is considered to be a "high" efficiency organization if the productivity index is greater than 10. During the simulation trials, the mean values for the productivity index is 10.08.

Assumption: PI **Cell: E6**

Weibull distribution with parameters:

Location	-3.60
Scale	15.11
Shape	3.615131262



Selected range is from 0.00 to +Infinity
Mean value in simulation was 10.08

Figure 25 Productivity Index Distribution.

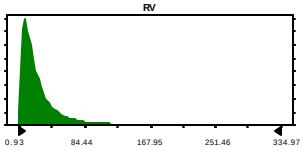
b. Requirements Distribution

The increase or decrease of requirements help derive the overall Requirements Volatility of the software project. A measure from the database can be negative or positive. Negative values indicate that a project experienced a reduction of requirements. For our purposes, the values normalized for to the absolute value. Figure 26 details how the requirements increase in projected. The mean value for the simulation was about 28% requirements volatility.

Assumption: RV **Cell: E5**

Lognormal distribution with parameters:

Mean	28.59
Standard Dev.	36.34



Selected range is from 0.00 to +Infinity
Mean value in simulation was 28.37

Figure 26 Requirements Growth Distribution.

c. E-SLOC Distribution

The Large Granular Complexity is a measure of the overall relational size of the software project [Dupo02]. For our simulations we are utilizing a lognormal distribution for the software project size in E-SLOC. Figure 27 details the uniform

distribution and identifies that our simulation trials experience a mean of 79 thousand effective source lines of code.

Assumption: E12 **Cell: E12**

Lognormal distribution with parameters:

Mean	79,129.70
Standard Dev.	132,961.36

Selected range is from 0.00 to +Infinity
Mean value in simulation was 78,611.39

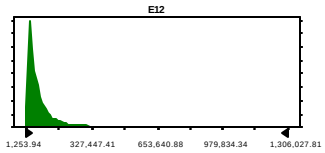


Figure 27 E-SLOC Distribution.

d. Actual Project Duration Distribution

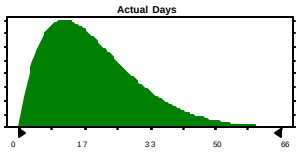
The project duration is recorded in the database in months. We convert these months to days by multiplying by 22 (working days). The 112 actual projects demonstrated a beta distribution with a mean value of 18 months or about 400 days.

Assumption: Actual Days **Cell: L15**

Beta distribution with parameters:

Alpha	1.98
Beta	9.80
Scale	111

Selected range is from 0 to +Infinity
Mean value in simulation was 18



Actual Project Duration Distribution

5. Software Risk Model Performance

With the establishment of the distribution assumptions, we can now focus our efforts on the simulation trials. Figure 24, the simulation architecture, illustrates that the each forecast should be compared with one another: software risk model and actual data. Each simulation assumption is to be exercised 10,000 trials. The next four illustrations present the *Actual Day* forecast on the left diagram and the *software risk model* forecast in the right diagram. Figure 28 illustrates the frequency distribution between the two forecast.

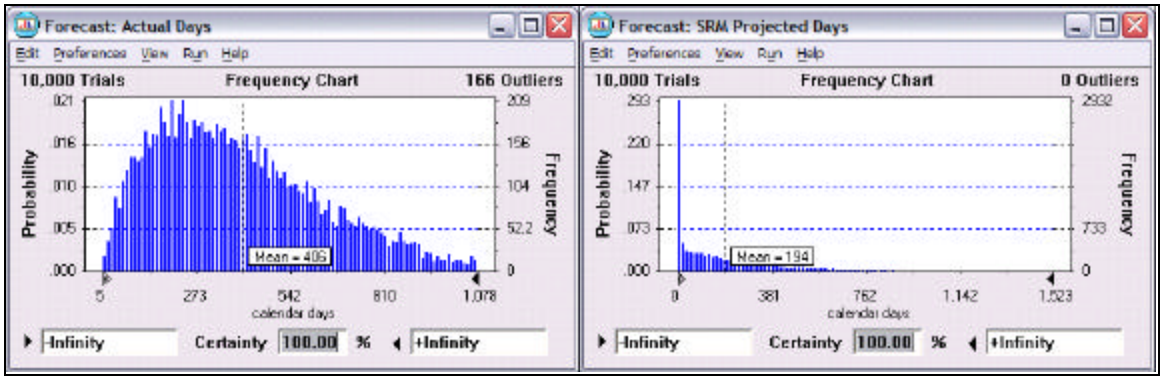


Figure 28 Actual vs Projected (100% Certainty).

The 112 projects were filtered down to 86 projects at this point. Although not documented, the software risk model is incapable of projecting software projects below 133 LGC (approximately 5500 E-SLOC). The distributions for the actual project days and E-SLOC were based off just 86 projects. The mean for the actual projects is 406 days. The risk model projected a mean of 194 days or about 50% less. This is a preliminary indication that the software risk model projects very optimistic schedules. The software risk model projects 84% of the projects complete prior to the mean of the actual data. Figure 29 is an overlay of both the frequency distribution and the cumulative distributions of the two forecast. Ideally, these two projection lines should exhibit minimal distance between the two lines. Unfortunately, this is not the case.

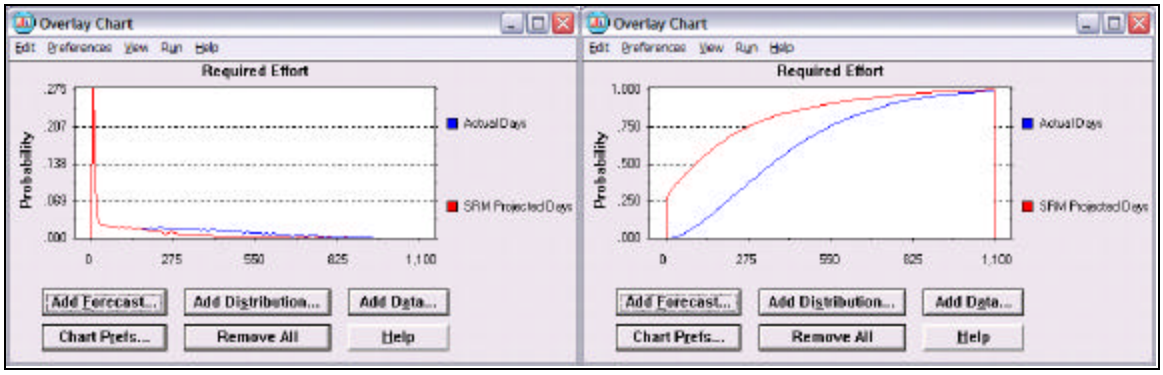


Figure 29 Actual vs Projected (Overlay Chart).

Figure 30 considers the two distributions after six months (132 days). Only 12.36% of the actual projects are complete while 54.76% of the software risk model

projects are complete; supporting our surjection of the software risk model projecting overly optimistic.

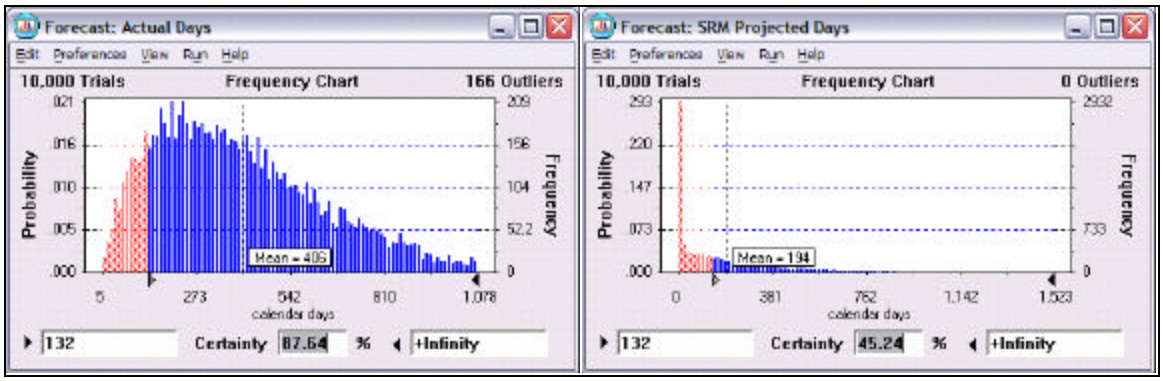


Figure 30 Actual vs Projected (6 Months).

Figure 31 demonstrates projections after 12 months. Thirty-three percent of the actual projects are completed at this time. However, in stark contrast, the software risk model has projected 73% of the software projects complete.

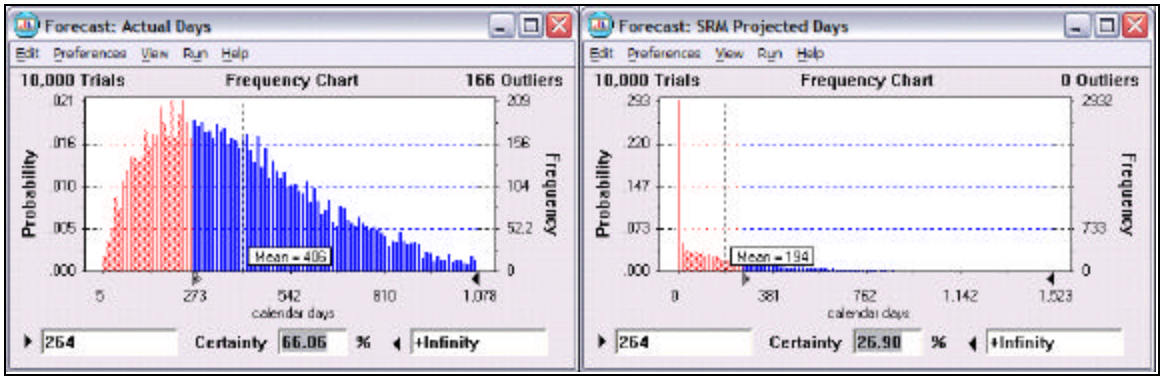


Figure 31 Actual vs Projected (12 Months).

Observations are made that in about 30 percent of the trials, the software risk model projected a dangerously low amount required time. This is after filtering out any projects under 5500 E-SLOC. There seems to be some problems with how the software risk model is implementing the projections.

VI. CONCLUSION

The software risk model developed by Dr. Juan Carlos Nogueira shows exceptional promise at delivering meaningful risk assessments early in the software development process. However, the model is unproven and remains invalidated outside of an academic environment. This thesis details a plan to thoroughly validate the software risk assessment model and provides for opportunity to extend and enhance the model when necessary.

> Nearly every software engineering development project is plagued with numerous problems leading to late delivery and cost overruns, and sometimes, unsatisfied customers. Often the problems are technical, but just as often the software engineering development problems are managerial. How often have we personally heard or read that a project was late (or over budget, or reduced in scope, or terminated early, or did not satisfy the user, …) because:
>
> - Programmers did not tell the truth (or did not know the truth) about the status of their programs
> - Top management did not allow sufficient time for upfront planning
> - The true status of the project was never known
> - Programmer productivity was lower than planner
> - The customer did not know what he wanted (or changed the requirements)
> - Government standards for requirement specification (or government policies) were not suitable for software
> - Or, or, or…

This quote from Richard Thayer is just as applicable today, if not more, than when he originally published it over twenty years ago in 1981 [Thay81]. Have we made any progress? I surmise that we have. Over the last forty years, research and technology have successfully overcome critical barriers to the advancement of the software engineering. Illustrated in Figure 32, computing power can be considered one of the earliest barriers we have overcome. Having the opportunity to access a computer and

having computers readily available soon joined computational power as issues of the past. The internet era of computing has spawned enormous success in the creation of efficient bandwidth and better networks. With all of the wonderful advances in technology, we still suffer from the inability to "… predict how much a software system will cost, when it will be operational, and whether or not it will satisfy user requirements …"[8].

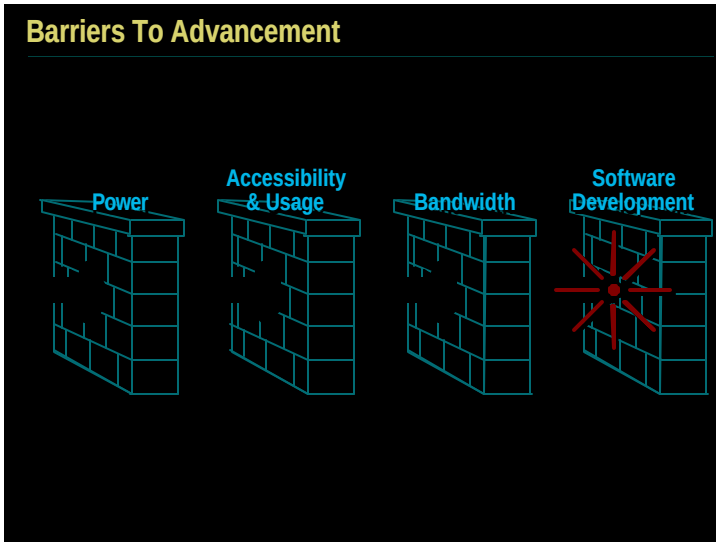


Figure 32 Barriers To Advance.

This thesis proposes a research opportunity advances the state-of-the art in software estimation. Significant enhancements can be made to the formal risk assessment model for software projects developed by [Nogu00]. The proposed research contained within validates the assertion that successful software risk assessment is achievable from a set of simple and easily derived artifacts. These artifacts can be automatically collected and are not subjective to an analyst's bias.

A. PROPOSED CONTRIBUTIONS OF THE DISSERTATION

- The proposed body of research could make the following advancements to the software engineering state of the art.
- A validated software risk model. This is the primary research thread of the research and can be achieved.

[8] Keynote speaker address from Lloyd K Mosemann II, Software Technology Conference, April 29, 2002.

- Interface techniques between the input parameters: efficiency, requirements, functional complexity, and functional size. Extensive analysis to provide suitable procedures to interface with information frequently collected in actual project development.
- A new technique to calibrate VitéProject to represent software development. Previously, VitéProject had never been calibrated to project software development. The proposed research provides techniques to utilize a calibrated VitéProject for accurate software estimations.
- Baseline software development trends. The research will provide a recommended baseline to extend or develop any future software development models; or to tune additional simulations.
- Possibly evidence to suggest the software risk model, in its original form, is not suitable to project software risk assessment. This thesis documents, unequivocally, the mathematical implementation of the software risk model is not correct. However, we do support the theoretical basis of the software risk model; that software risk assessment can be achieved with analysis of a small set of easily obtainable, mathematically quantifiable, software metrics.

B. POTENTIAL AREAS FOR FUTURE WORK

As much as we would like to think, no software estimation model is a silver bullet. Software engineering and software risk assessment are a long way from making the opening quote from Richard Thayer obsolete. Although the proposed research provides a suitable mitigation to approaching software development blind and naive, there exist several areas where future work is warranted.

- Any model must continue to be validated. Our validation is conducted against 2000 projects. However, these 2000 projects were comprised of ten application types. Increasing the numbers of projects in each application type could serve two purposes (1) provides additional data points to project trend lines, and (2) provides additional data points to test the different models.
- Due to resource constraints, we do not propose to challenge [Nogu00]'s interpretation of the organizational efficiency or the requirements volatility. In their current forms, these two parameters will seem to suffice. However, there is always room for improvement. I would first recommend revisiting the collection techniques of these two parameters.
- Any useful model will have to be calibrated. It is very reasonable to appreciate technologies change and software developers will always provide better techniques and practices to develop software solutions. Complex software today may or may not be complex tomorrow.

However, a model that does not have a capability to extend and remain current minimizes its long-term sustainability.

- A PSDL parser must be developed to implement the excellent work in [Dupo02]. The parser, or direct implementation into CAPS, will provide an automated means to derive the complexity of the software prototype.
- Additional projects must be developed in the CAPS environment. The available data points from completed software project are very limited. Additional projects would permit the continued development of the complexity measure in [Dupo02] and provide empirical evidence to develop the translation of the PSDL complexity measure to the MRM functional complexity.
- The software risk model, or the proposed research, has yet to be implemented on a project from ground zero. All model validations are conducted on post-mortem or simulated projects.

APPENDIX A. DATA DICTIONARY

This appendix provides the details of the metrics that have been collected in the Quantitative Software Management (QSM®) database. These are the metrics that are available to conduct my research against. It should be noted that corporations using the database have the opportunity to work with a set of default metrics, as well as collect information specialized to the corporation's particular needs. In the following, the default metrics are presented. This data dictionary of metrics is taken directly from a QSM® publication [QSMa].

A. BASIC INFORMATION

Project Name. System name or project title.

Status. Indication of the current state of data entered for the project; choose one of the following:

Estimate – The collected data are estimated.

In Progress – The collected data contain partial actuals.

Completed – The collected data contains final actuals.

Confidence. Label that best describes the level of confidence in the accuracy and consistency of this project's data, determined from one of the following:

Low – Low Confidence (error > 10%)

Moderate – Moderate Confidence (5% > error < 10%)

High – High Confidence (error < 5%)

Record Creation Date. Date this project data was first entered.

Last Modified. Date this project data was last modified. *Note, in the extract of the database used for my research, date of last modified is determined by the date the company names were removed.*

Predominant Application Type and Sub-type

Label that best describes the application type most heavily represented in this system. There are nine primary application types and one or more subtypes within each

of these nine types. First find the primary type that coincides with your typical project, according to the following descriptions.

Microcode & Firmware. Software that is either the architecture of a new piece of hardware or software that is burned into silicon and delivered as part of a hardware product. This software is the most complex because it must be compact, efficient, and extremely reliable.

Real Time. Software that must operate close to the processing limits of the CPU. This is interrupt driven software and is often written in C, Ada or Assembly language. Typical examples are military systems like radar, signal processors, missile guidance systems, etc.

Avionics. Software that is on-board and controls the flight and operation of the aircraft.

System Software. Layers of software that sit between the hardware and applications programs. Examples are operating systems (DOS, UNIX, VMS, etc.), GUI's (graphical user interfaces – Windows, Xwindows etc.), Executives or Database Management systems, Network products, and Image processing products.

Command & Control. Software that allows humans to mange a dynamic situation and respond in human real-time. Examples are battle field command systems, telephone network control systems, government disaster response systems, military intelligence systems, electric utility power control systems.

Telecommunications. Software that facilitates the transmission of information from one physical location to another. Examples are telephone switches, transmission systems, modem communication products, fax communication products, satellite communications products.

Scientific. Software that involves significant computations and analysis. Examples are statistical analysis systems, graphics products, data reduction systems.

Process Control. Software that controls an automated system. Examples are software that runs a nuclear power plant, or software that runs a petro-chemical plant.

Business. Software that automates a common business function. Examples are payroll, personnel, order entry, inventory, materials handling, and warranty products.

Description. Overview of the system under development.

Size. The function unit selected for sizing. If this is a new project, the companies default size will appear, otherwise, the primary unit will be the first unit for which sizing data has been entered.

New. Portion, measured in the selected size units, of the total system size that was developed for this application (designed, coded, integrated, and tested from scratch).

Modified. Portion, measured in the select size units, of the total system size contained in pre-existing entities (modules, packages, etc.) that were modified (pre-existing software requirement changes).

Unmodified. Portion, measure in the selected size units above, of the total system size contained in pre-existing entities (modules, packages, etc.) that were incorporated into this product unchanged (pre-existing software requirement no change)

Requirements. Total number of requirement defined during the Critical Design Review.

Defects System Integration to Delivery. Total number of defects found between SIT and the completion of the main build. SIT (System Integration Test) is a QSM-defined milestone that represents the point at which a configuration item (e.g., package, compilation unit) is placed under change control and is ready to be integrated into the evolving system.

Defects First Month after Delivery. Total number of defects found during the first month following FOC. FOC (Full Operational Capability) is a QSM-defined milestone that represents full functionality with 95% reliability.

Life Cycle Values. The following information is provided for each phase of development.

Phase Acronym. The acronym, if any, that the project uses to refer to the activities of the associated life cycle phase.

Start Date. The date on which effort was first applied to any activity of the associated life cycle phase.

End Date. The date on which all activities of the associated life cycle phase were completed.

Months. The elapsed number of months for this phase.

Effort. Total amount of effort in the unit specified of labor to complete all activities of the associated life cycle phase. Includes all development staff: analysts, designers, programmers, coders, integration and test-team members, quality assurance, documentation, supervision, and management.

Cost. Total cost, in thousands of monetary units, to complete all activities of the associated life cycle phase.

Peak Staff. Peak number of full time equivalent people used to complete all activities of the associated life cycle phase.

Staffing Shape-Phases 1,2,3. Label that best describes the shape of the staffing curve for the associated life cycle phase:

Unknown – Valid for any of the four life cycle phases.

Level Load – A staffing plan that has the same number of people, or essentially the same number, from beginning to end. Usually found in the development of small systems.

Front Load Rayleigh – The front load Rayleigh shape will peak about 40% of the way through the phase and then taper down.

Medium Front Load Rayleigh. The medium front load Rayleigh profile will peak close to the middle of the phase and then start to taper down in staffing.

Medium Rear Load Rayleigh. The medium rear load Rayleigh peaks about three quarters of the way through the phase.

Rear Load Rayleigh. Peaks at the end of phase 3.

Default Rayleigh. A roughly bell shaped curve used to represent the ideal buildup and decline of manpower, effort, or cost, followed by a long tail representing manpower, effort, or cost devoted to enhancement or maintenance.

Front Weibull. The front load Weibull shape will peak about 25% of the way through the phase and then taper down.

Normal. The normal shape peaks at about midway through the phase and then tapers down.

Rear Weibull. The real load Weibull shape peaks about 70% of the way through the phase and then taper down.

Custom. Any other staffing pattern not described by the other options presented.

Staffing Shape-Phase 4. Label that best describes the shape of the staffing curve for the associated life cycle phase:

Stair Step. Staffing starts at about half of the staffing level at FOC and stair-steps down.

Straight Line. A straight line decrease in staffing.

Exponential. One of the phase 4 staffing patterns, represented by the fastest tailing off of people from milestone 7 to milestone 9 [QSMb].

Rayleigh. Tail-off of Rayleigh curve selected in phase 3. Valid only if phase 3 staffing is a Rayleigh shape.

Normal. The normal shape peaks at about midway through the phase and then tapers down.

Weibull. Tail-off of Weibull curve selected in phase 3.

Custom. Any other staffing pattern not described by the other options presented.

Life Cycle Semantics.

Phase 1: Acronym FEAS, Label Feasibility Study. The earliest phase in the software life cycle, where complete and consistent requirements and top-level, feasible plans for meeting them are developed.

Phase 2: Acronym FUNC, Label Functional Design. A phase of the software development process prior to phase 3 that develops a technically feasible, modular design with the scope of the system requirements.

Phase 3: Acronym MB, Label Main Build. A phase in the software development process which produces a working system that implements the system specifications and meets system requirements in terms of performance and reliability.

Phase 4: Acronym MAINT, Label Maintenance. The phase that usually coincides with the operations phase. It may include correcting errors that operations turn up and enhancing the system to accommodate new user needs, to adapt to environmental changes, and to accommodate new hardware.

B. APPLICATION

Organization. The name of the company or organization entity that was responsible for the project. *Due to the sensitive nature of the contents of the database, company names have been removed.*

Division. The name of the organization's particular business unit that was responsible for the project.

Country. The name of the country in which the project was done.

Design Complexity. Label that best describes the complexity of the developed system (Low, Medium, High)

Development Classification. Label that best describes the type of development effort undertaken by the project.

Brand New System - > 75% new function

Major Enhancement – 25 to 75% new function

Minor Enhancement – 5 to 25% new function

Conversion - < 5% new function

Industry Sector

The Industry/Sector field is presented in a treed list.

Figure 33 presents the sector description with an industry that best represents the majority of projects in your organization.

Industry	Sectors
Academic	(General)
Aerospace	(General Aviation, Commercial, Military)
Construction	(General)
Distribution	(General)
Military	(General)
Electronics	(General)
Engineering	(General)
Financial	(General, Banking, Insurance)
Government	(General, National, Local, State, County)
Leisure	(General, Entertainment, Hotel)
Manufacturing	(General, Airplane, Automobile, Brewery, Chemical, Computer, Comp Peripherals, Image Processing, Medical, Pharmaceutical, Software, Telecom Equipment)
Health	(General)
Mining	(General, Metals, Gem, Coal)
Oil	(General)
Retail	(General, Food, Clothing)
Service	(General, Consulting)
Transportation	(General, Air, Bus, Rail/Metro, Sea, Trucking)
System Integrators	(General)
Utilities	(General, Gas, Electric, Telecom, Water)
Other	(General)

Figure 33 Industry Sectors.

Application Type. The percentage of the total system size dedicated to the associated application type.

Development Machine Type. The general type of the host development machine.

Development Machine Specific. The specific name and version of the host development machine.

Operating System Type. The general type of the host operating system.

Operating System Specific. The specific name and version of the host operating system.

C. SIZING

Function Unit. The name of a sizing unit used to size your software. If you are not able to find the appropriate size measure, you may add a new unit .

Counting Method. The method used for counting the associated sizing unit:

Manual – Physically counting the source code by hand

Estimated – An educated guess, perhaps based on analogy from past experiences

Sampled - Using small representative sample statistics to forecast the complete system size

Code Counter – An automated code counter

Gearing Factor. The factor that, when multiplied by the total of units you will enter, yields the total system size in logical Source Lines of Code (SLOC).

Language. The name of a language used to code the system.

% of Total Size. The percentage of the total system size accounted for by the associated language.

Language Type. Select the language type that best describes the level of the associated language.

D. ACCOUNTING

Effort Units. An effort unit allows you to select man years, man months, man days or man-hours. They have a first character designation of "M". You may select person years, person months, person days or person hours (designation "P"), or you may select from staff years, staff months, staff days, staff hours (designation "S"). There is no difference between a man-year, person year, or staff year. It just provides you the flexibility of selecting the gender and nomenclature that's most appropriate in your organization.

Person Hours per Person Month. The average number of hours a typical Full Time Equivalent (FTE) employee works in a 30.4375-day month allowing for vacation, holidays, and paid personal time off. The value should range between 120 and 744.

Labor Rate. The average wage and salary rate of the people directly involved in a software development, including detailed design, coding, testing, validation, documentation, supervision, and management, plus a percentage of the direct average to account for overhead.

Labor Rate Unit. The name of the effort unit associated with the given Labor Rate; choose one the following: MYR, MM, MWK, MDAY, MHR, PYR, PM, PWK, PDAY, PHR, SYR, SM, SWK, SDAY.

Monetary Unit. In the monetary units drop down panel, you may select a code for the currency to be used on this project.

Conversion Factor (to $US). The factor that, when multiplied by an amount expressed in the selected Monetary Units, yields an equivalent amount in United States dollars.

E. ENVIRONMENT

Development Environment. The default development environment is where you specify the environment in which your software was developed. In most cases, this will be 5 days a week, 8 hours a day, 60 minutes per hour and 60 seconds per minute.

Operational Environment. The default operational environment is where you specify the environment in which your software will run.

F. QUALITY

Days/Week. The number of days per week that the runtime environment was up and running.

Hours/Day. The number of hours per day that the runtime environment was up and running.

Minutes/Hour. The number of minutes per hour that the runtime environment was up and running.

Seconds/Minute. The number of seconds per minute that the runtime environment was up and running.

Defect Categories. The labels of the 5 defect categories.

Cosmetic Defects. The name that corresponds to QSM's cosmetic defects. Cosmetic defects can be described as deferred, such as errors in format of displays or printouts. They should be fixed for appearance reasons, but fix may be delayed until convenient.

Tolerable Defects. The name that corresponds to QSM's tolerable defects. Tolerable defects can be described as ones that do not affect the correctness of outputs; they should be fixed but may be delayed until convenient.

Moderate Defects. The name that corresponds to QSM's moderate defects. Moderate defects can be described as not critical to execution of the program but behavior is only partially correct. Should be fixed in the release.

Critical Defects. The name that corresponds to QSM's critical defects. Critical defects can be described as ones that prevent further execution; unrecoverable; they should be fixed before the program is used again.

G. REVIEW (PROJECT OVERRUNS)

Time (months). Planned time in calendar months (or % difference from plan) for the specified phase. A planned time less than the actual time will result in a positive overrun %. A negative % overrun indicates early completion.

Effort. Planned labor in the indicated effort unit (or % difference from plan) for the specified phase.

Cost. Planned cost in thousands of the indicated monetary unit (or % difference from plan) for the specified phase.

Peak Staff. Planned peak staff in number of full time equivalent people (or % difference from plan) for the specified phase. (Calculated by dividing the total effort in person months by the total time in months for a given phase.)

Size and Requirements Growth. How much did the project change from the original plan, either the percentage of the original planned value or as the actual planned value.

System Benefit / Effectiveness. An indication of how beneficial or effective the final system was in solving the problem it was intended to solve. If the system was a commercial product, then the effectiveness should be interpreted as its commercial success.

Significant Factors. Describe any factors that had a positive or a negative impact on the project.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INITIAL PROJECT OVERVIEW

A. OVERVIEW OF DATABASE

It is important for readers to understand the quality of the information obtained in the subset of projects available for conducting research. This appendix includes extracts from the initial set of 112 projects used in the research. The projects were chosen based on their similarities with other projects already exercised against Nogueira's risk assessment model [Nogu00 and John01]. The following is an overview of the subset of projects and a limited number of extracts on the actual data.

Figure 34 details how the total number of projects are categorized according to the application type. The majority of the projects are from the Avionics industry while the least number of projects represent Microcode and Real-time systems.

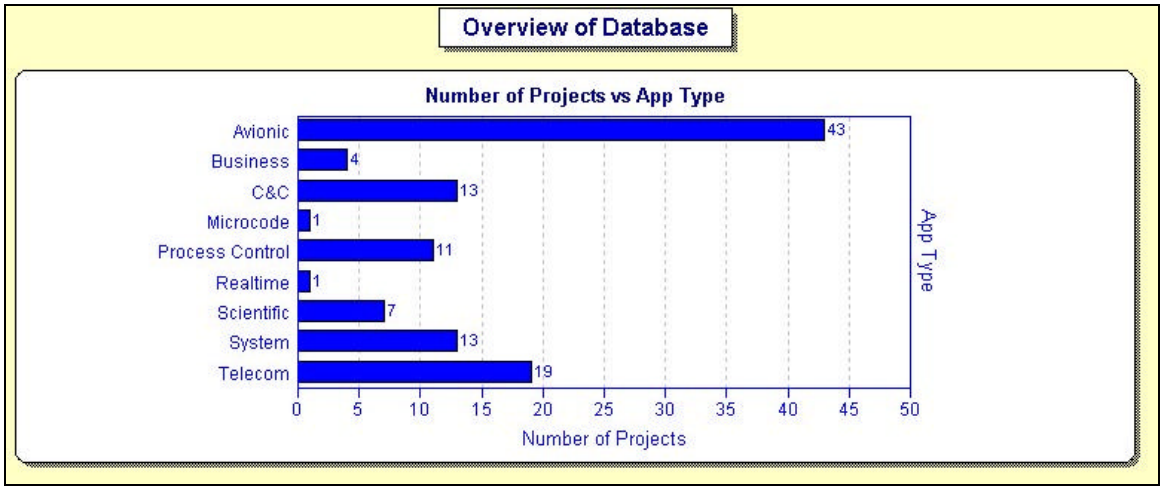


Figure 34 Projects vs. Application Type.

In an effort to preserve proprietary information, a generic identifier has replaced organization names. In the subset of initial projects available to evaluate, there are a total of 26 different organizations. Figure 35 displays the number of projects captured in the database by each organization.

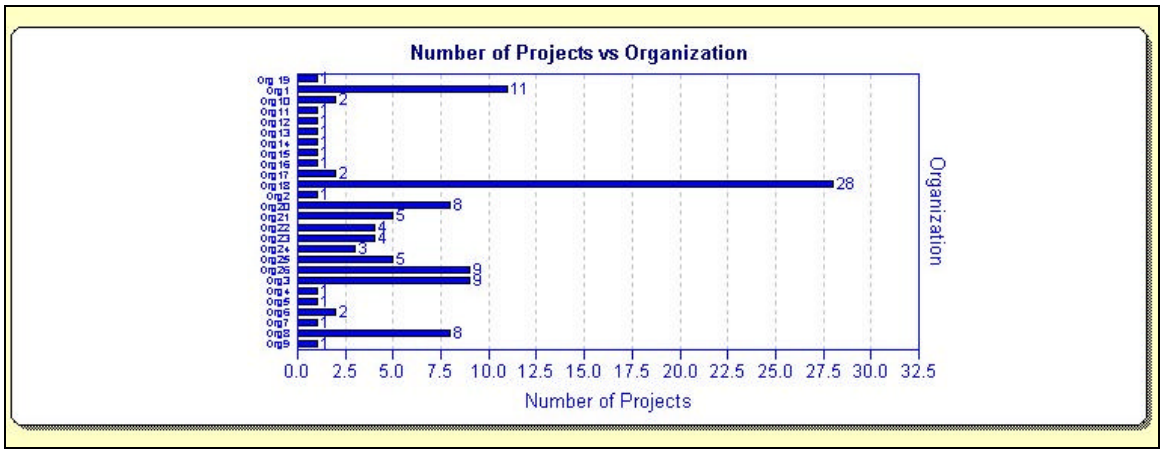


Figure 35 Number of Projects per Organization.

Organizations represented in the database, use a metric that is called a "productivity index" or PI. The PI of an organization, for a particular project, is a management scale from one to 36, corresponding to the productivity parameter, that represents the overall process productivity achieved by an organization during the main build [Putn92]. The PI is a measure that quantifies the net effect of everything that makes projects different from one another.

In the database under consideration, the majority of projects were developed by organizations with a PI between nine and eleven, see Figure 36. Two projects have been developed by organizations with a very low PI and one project has been developed by an organization with a PI between twenty-two and twenty-three. It should be noted that an increase in one PI yields approximately a 10% reduction in schedule and approximately a 25 – 30% reduction in Effort. The typical improvement rate is approximately one PI per 2 – 2.5 years; similar to the normal time required moving up one level of the SEI CMM [QSMc].

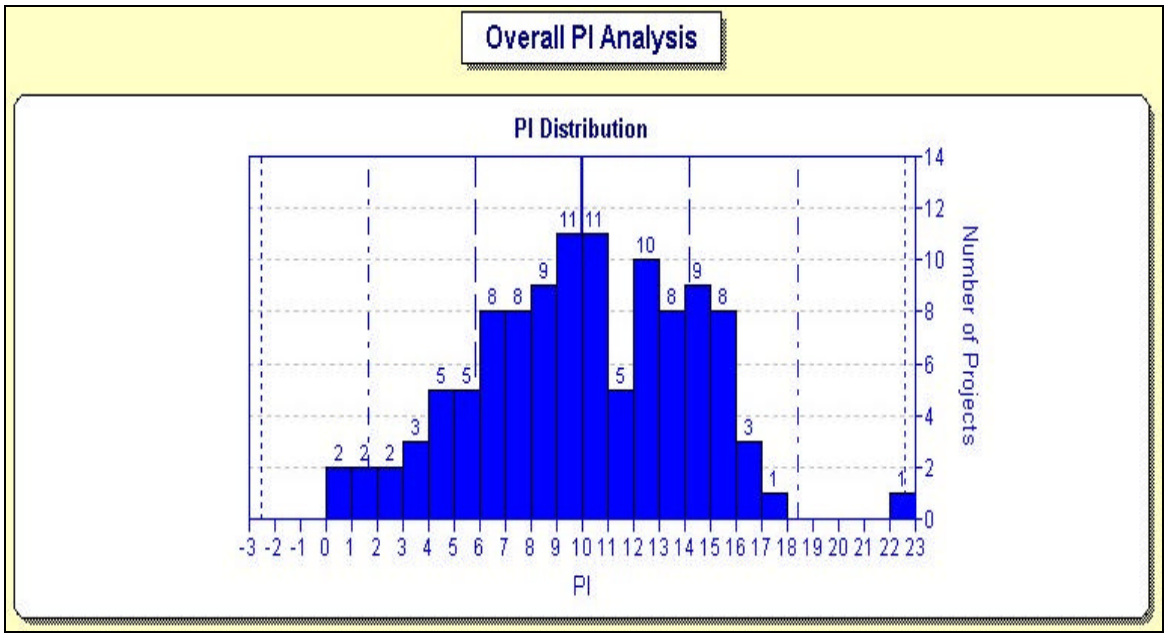


Figure 36 Overall PI Analysis.

Figure 37 shows the 26 organizations in the database and their corresponding PI, in terms of an average, minimum, and maximum. For example, Organization 8 developed projects with a minimum PI of approximately 4.9 and a maximum PI of approximately 22.9. The average PI recorded for Organization 8 is approximately 9.55.

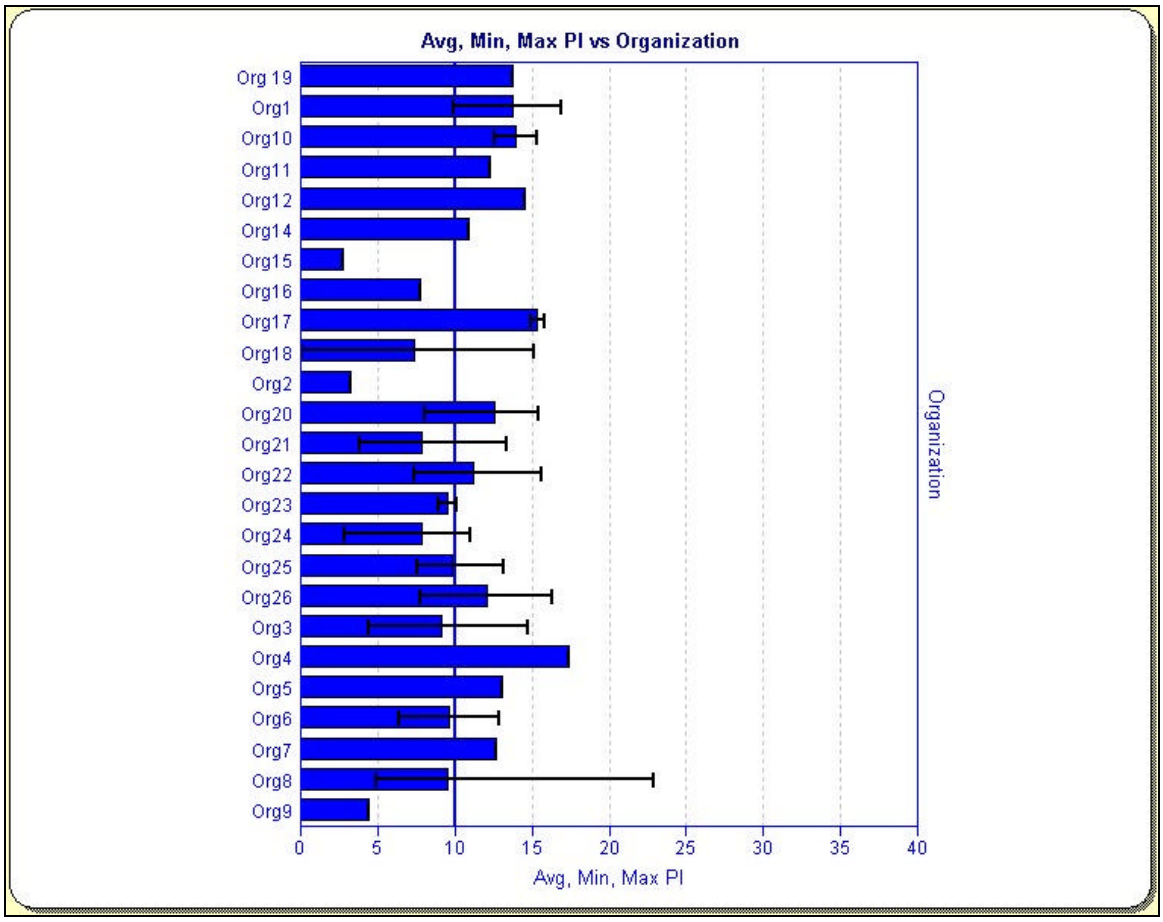


Figure 37 Organizational PI Distribution.

In Figure 38, an indication of the average staff months required during all four phases of project development is illustrated. In this view, the Main Build took an average of 270 staff months to complete.

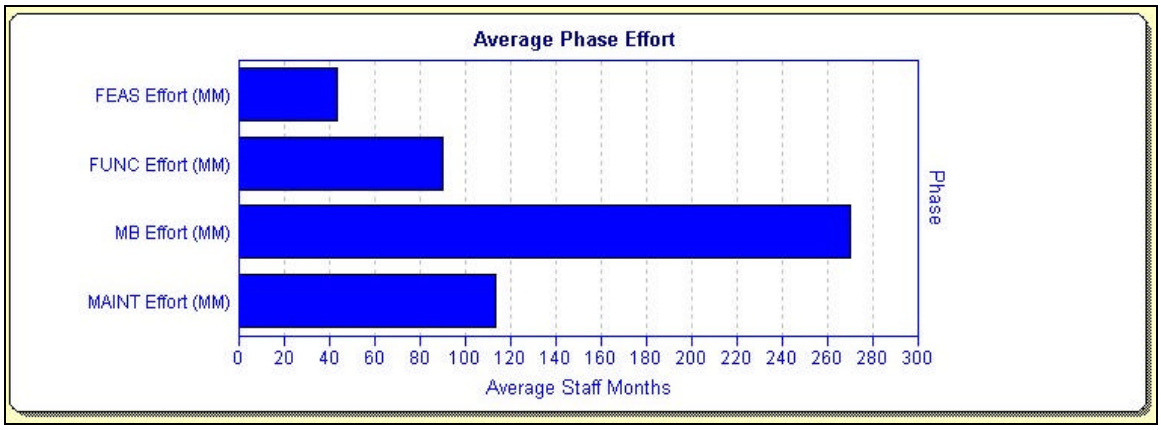


Figure 38 Average Phase Effort.

To further clarify the effort expended to produce the software projects, Figure 39 considers the actual expended calendar time to complete each software development phase.

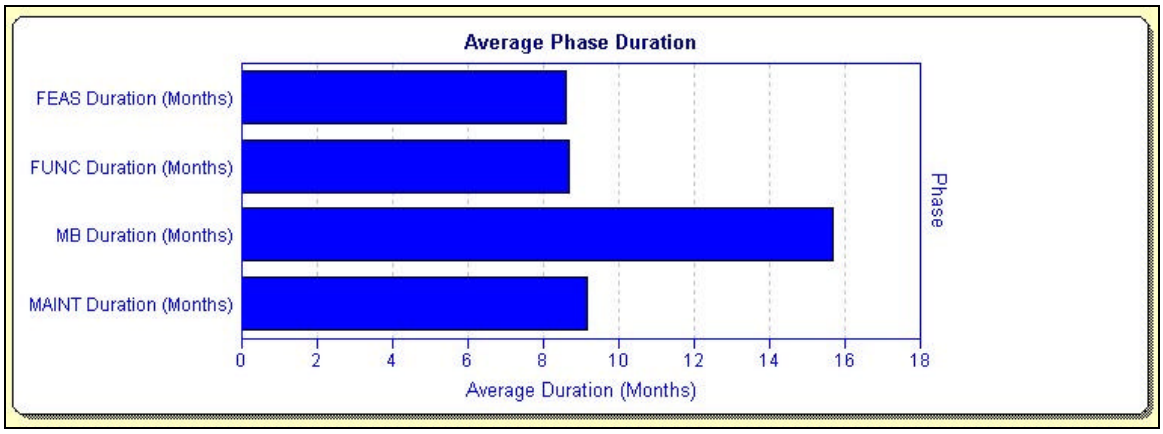


Figure 39 Average Phase Duration.

Figure 40 shows four views to help the reader understand trends in the size of the software projects under consideration. Without consideration for the staffing size, the top left chart compares the size of the projects (in source lines of code) against the calendar time expended constructing the project. The top right chart does consider the staffing size and completes a similar comparison between the project size and effort.

Staffing sizes are captured in the database by various staffing profiles. The bottom left chart expresses the project size against the peak staff while the bottom right chart expresses the project size against the average staff.

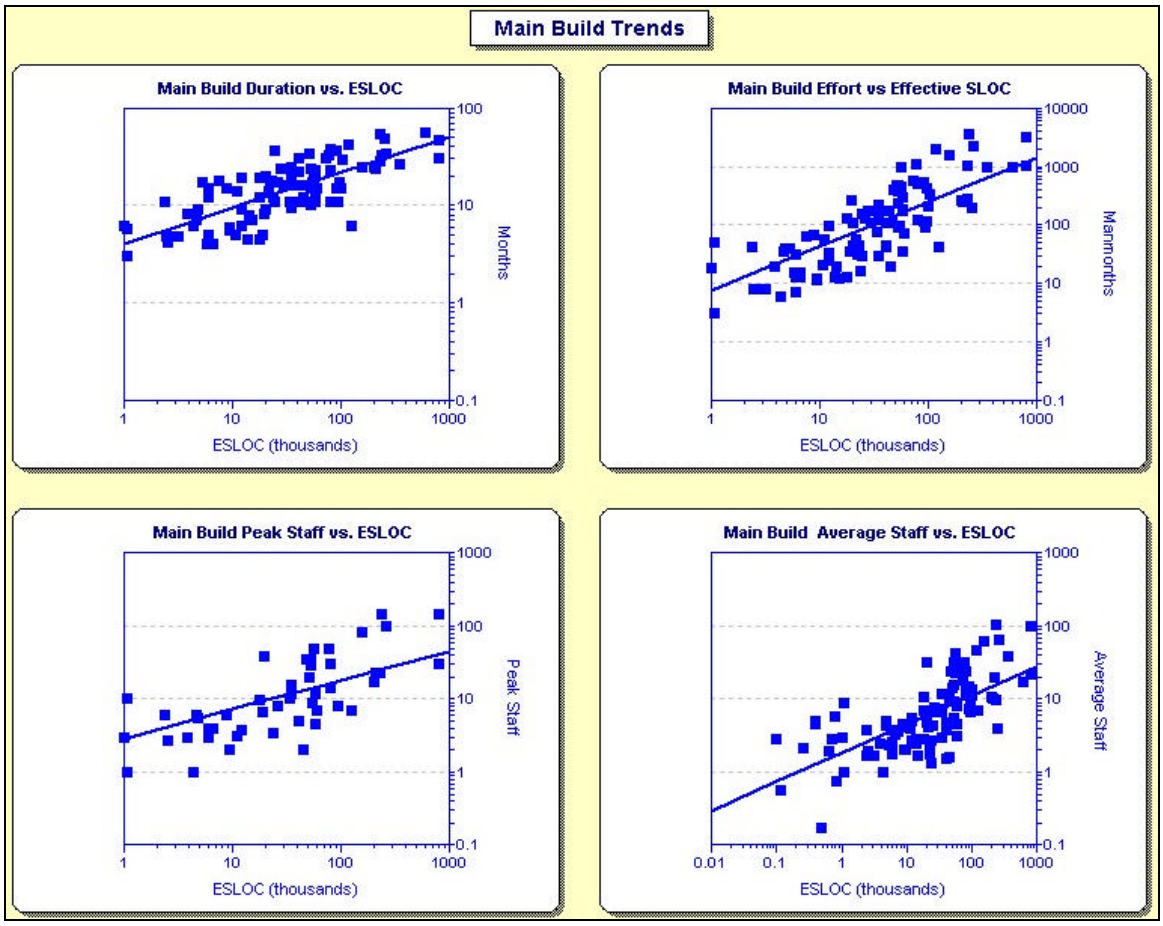


Figure 40 Main Build Trends.

Finally, it is prudent to express how schedule slippages are captured in the database, see Figure 41. For example, during the main build, 33% of all of the projects ran over their initial estimation by as much as 25%. Probably more alarming is the fact that out of the 112 projects under consideration, only 13% finished on time or earlier that their schedule estimates.

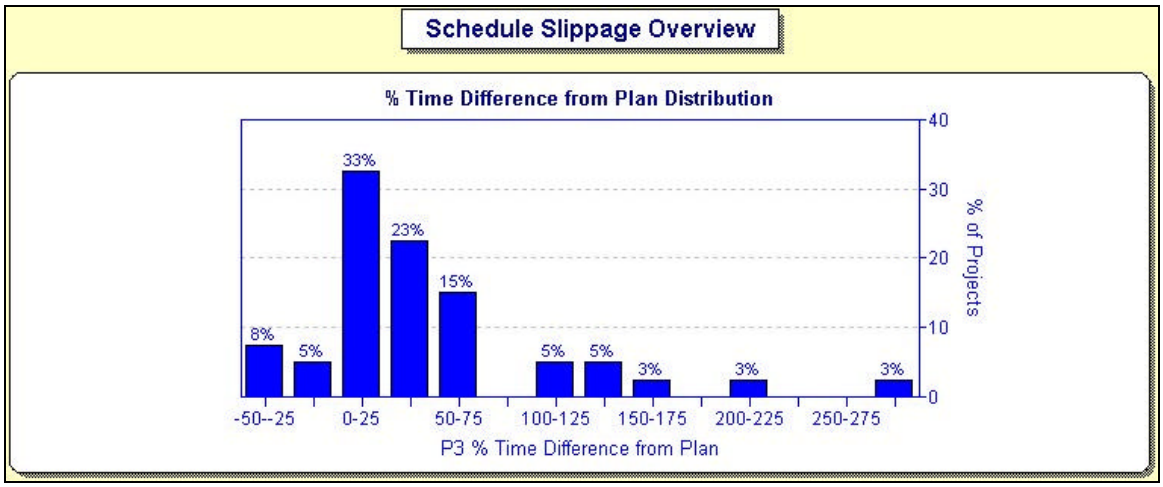


Figure 41 Schedule Slippage Overview.

B. SPECIFIC SOFTWARE PROJECT EXTRACTS

The next section illustrates some actual data as it is represented in the project database. Data is included from four projects. The difference in these four projects is the amount of information that is captured on each. The minimum set of data representing a project is data from the main build. The maximum set of data collected is data collected from all four phases of the project life cycle.

Presently, the research is not mature enough to depict the potential impact of not having collected data during all four phases of the project life cycle. Nogueira's model is intended to derive the probability of project completion from the requirements through the delivery. A seemingly logical hypothesis is that it is necessary to only consider projects with data represented to cover this time span of project development.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SUPPORT DATA FOR SIMULATION

The following are two reports generated from the Monte Carlo simulation software Crystal Ball. These reports directly support the implementation of Chapter V.

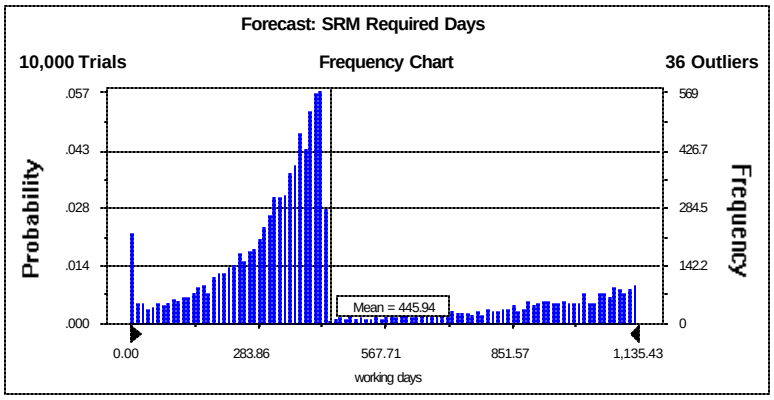
A. SOFTWARE RISK MODEL BEHAVIOR

Forecast: SRM Required Days **Cell: L14**

Summary:

Display Range is from 0.00 to 1,135.43 working days
Entire Range is from 0.00 to 1,141.87 working days
After 10,000 Trials, the Std. Error of the Mean is 2.84

Statistics:	Value
Trials	10000
Mean	445.94
Median	378.72
Mode	---
Standard Deviation	283.95
Variance	80,628.34
Skewness	1.07
Kurtosis	3.27
Coeff. of Variability	0.64
Range Minimum	0.00
Range Maximum	1,141.87
Range Width	1,141.87
Mean Std. Error	2.84



Forecast: SRM Required Days (cont'd) **Cell: L14**

Percentiles:

Percentile	working days
0%	0.00
10%	164.75
20%	254.01
30%	311.13
40%	348.99
50%	378.72
60%	402.88
70%	423.39
80%	665.28
90%	976.34
100%	1,141.87

Frequency Counts:

Frequency:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	0.00	0.000000	0
1	0.00	11.35	0.022300	223
2	11.35	22.71	0.005300	53
3	22.71	34.06	0.004800	48
4	34.06	45.42	0.003500	35
5	45.42	56.77	0.004100	41
6	56.77	68.13	0.004800	48
7	68.13	79.48	0.004400	44
8	79.48	90.83	0.005100	51
9	90.83	102.19	0.006200	62
10	102.19	113.54	0.005800	58
11	113.54	124.90	0.006600	66
12	124.90	136.25	0.006300	63
13	136.25	147.61	0.007200	72
14	147.61	158.96	0.008900	89
15	158.96	170.31	0.009400	94
16	170.31	181.67	0.007700	77
17	181.67	193.02	0.011400	114
18	193.02	204.38	0.012300	123
19	204.38	215.73	0.012300	123
20	215.73	227.09	0.014100	141
21	227.09	238.44	0.014400	144
22	238.44	249.79	0.017300	173
23	249.79	261.15	0.015700	157
24	261.15	272.50	0.017600	176

Forecast: SRM Required Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
25	272.50	283.86	0.017900	179
26	283.86	295.21	0.020900	209
27	295.21	306.57	0.023600	236
28	306.57	317.92	0.026700	267
29	317.92	329.27	0.031000	310
30	329.27	340.63	0.031000	310
31	340.63	351.98	0.031500	315
32	351.98	363.34	0.036800	368
33	363.34	374.69	0.038800	388
34	374.69	386.05	0.046700	467
35	386.05	397.40	0.043000	430
36	397.40	408.75	0.052200	522
37	408.75	420.11	0.056800	568
38	420.11	431.46	0.056900	569
39	431.46	442.82	0.028000	280
40	442.82	454.17	0.000700	7
41	454.17	465.53	0.001100	11
42	465.53	476.88	0.001400	14
43	476.88	488.23	0.001200	12
44	488.23	499.59	0.001900	19
45	499.59	510.94	0.001200	12
46	510.94	522.30	0.001600	16
47	522.30	533.65	0.001200	12
48	533.65	545.01	0.001200	12
49	545.01	556.36	0.001900	19
50	556.36	567.71	0.001100	11
51	567.71	579.07	0.001600	16
52	579.07	590.42	0.002200	22
53	590.42	601.78	0.001300	13
54	601.78	613.13	0.002800	28
55	613.13	624.49	0.001800	18
56	624.49	635.84	0.001400	14
57	635.84	647.19	0.002400	24
58	647.19	658.55	0.001900	19
59	658.55	669.90	0.002200	22
60	669.90	681.26	0.002200	22
61	681.26	692.61	0.001900	19
62	692.61	703.97	0.002900	29
63	703.97	715.32	0.002400	24
64	715.32	726.67	0.003300	33
65	726.67	738.03	0.002900	29
66	738.03	749.38	0.002700	27

Forecast: SRM Required Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
67	749.38	760.74	0.002700	27
68	760.74	772.09	0.002300	23
69	772.09	783.45	0.003000	30
70	783.45	794.80	0.002300	23
71	794.80	806.15	0.003500	35
72	806.15	817.51	0.003300	33
73	817.51	828.86	0.003300	33
74	828.86	840.22	0.003600	36
75	840.22	851.57	0.003500	35
76	851.57	862.93	0.004200	42
77	862.93	874.28	0.003200	32
78	874.28	885.63	0.003500	35
79	885.63	896.99	0.005400	54
80	896.99	908.34	0.004500	45
81	908.34	919.70	0.005000	50
82	919.70	931.05	0.005400	54
83	931.05	942.41	0.005800	58
84	942.41	953.76	0.005300	53
85	953.76	965.11	0.005100	51
86	965.11	976.47	0.005400	54
87	976.47	987.82	0.005100	51
88	987.82	999.18	0.005100	51
89	999.18	1,010.53	0.004900	49
90	1,010.53	1,021.89	0.007300	73
91	1,021.89	1,033.24	0.004900	49
92	1,033.24	1,044.59	0.005100	51
93	1,044.59	1,055.95	0.007400	74
94	1,055.95	1,067.30	0.007500	75
95	1,067.30	1,078.66	0.006400	64
96	1,078.66	1,090.01	0.008900	89
97	1,090.01	1,101.37	0.008600	86
98	1,101.37	1,112.72	0.007300	73
99	1,112.72	1,124.07	0.008400	84
100	1,124.07	1,135.43	0.009500	95
	1,135.43	+Infinity	0.003600	36
	Total:		1.000000	10000

Cumulative:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	0.00	0.000000	0
1	0.00	11.35	0.022300	223
2	11.35	22.71	0.027600	276

Forecast: SRM Required Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
3	22.71	34.06	0.032400	324
4	34.06	45.42	0.035900	359
5	45.42	56.77	0.040000	400
6	56.77	68.13	0.044800	448
7	68.13	79.48	0.049200	492
8	79.48	90.83	0.054300	543
9	90.83	102.19	0.060500	605
10	102.19	113.54	0.066300	663
11	113.54	124.90	0.072900	729
12	124.90	136.25	0.079200	792
13	136.25	147.61	0.086400	864
14	147.61	158.96	0.095300	953
15	158.96	170.31	0.104700	1047
16	170.31	181.67	0.112400	1124
17	181.67	193.02	0.123800	1238
18	193.02	204.38	0.136100	1361
19	204.38	215.73	0.148400	1484
20	215.73	227.09	0.162500	1625
21	227.09	238.44	0.176900	1769
22	238.44	249.79	0.194200	1942
23	249.79	261.15	0.209900	2099
24	261.15	272.50	0.227500	2275
25	272.50	283.86	0.245400	2454
26	283.86	295.21	0.266300	2663
27	295.21	306.57	0.289900	2899
28	306.57	317.92	0.316600	3166
29	317.92	329.27	0.347600	3476
30	329.27	340.63	0.378600	3786
31	340.63	351.98	0.410100	4101
32	351.98	363.34	0.446900	4469
33	363.34	374.69	0.485700	4857
34	374.69	386.05	0.532400	5324
35	386.05	397.40	0.575400	5754
36	397.40	408.75	0.627600	6276
37	408.75	420.11	0.684400	6844
38	420.11	431.46	0.741300	7413
39	431.46	442.82	0.769300	7693
40	442.82	454.17	0.770000	7700
41	454.17	465.53	0.771100	7711
42	465.53	476.88	0.772500	7725
43	476.88	488.23	0.773700	7737
44	488.23	499.59	0.775600	7756

Forecast: SRM Required Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
45	499.59	510.94	0.776800	7768
46	510.94	522.30	0.778400	7784
47	522.30	533.65	0.779600	7796
48	533.65	545.01	0.780800	7808
49	545.01	556.36	0.782700	7827
50	556.36	567.71	0.783800	7838
51	567.71	579.07	0.785400	7854
52	579.07	590.42	0.787600	7876
53	590.42	601.78	0.788900	7889
54	601.78	613.13	0.791700	7917
55	613.13	624.49	0.793500	7935
56	624.49	635.84	0.794900	7949
57	635.84	647.19	0.797300	7973
58	647.19	658.55	0.799200	7992
59	658.55	669.90	0.801400	8014
60	669.90	681.26	0.803600	8036
61	681.26	692.61	0.805500	8055
62	692.61	703.97	0.808400	8084
63	703.97	715.32	0.810800	8108
64	715.32	726.67	0.814100	8141
65	726.67	738.03	0.817000	8170
66	738.03	749.38	0.819700	8197
67	749.38	760.74	0.822400	8224
68	760.74	772.09	0.824700	8247
69	772.09	783.45	0.827700	8277
70	783.45	794.80	0.830000	8300
71	794.80	806.15	0.833500	8335
72	806.15	817.51	0.836800	8368
73	817.51	828.86	0.840100	8401
74	828.86	840.22	0.843700	8437
75	840.22	851.57	0.847200	8472
76	851.57	862.93	0.851400	8514
77	862.93	874.28	0.854600	8546
78	874.28	885.63	0.858100	8581
79	885.63	896.99	0.863500	8635
80	896.99	908.34	0.868000	8680
81	908.34	919.70	0.873000	8730
82	919.70	931.05	0.878400	8784
83	931.05	942.41	0.884200	8842
84	942.41	953.76	0.889500	8895
85	953.76	965.11	0.894600	8946
86	965.11	976.47	0.900000	9000

Forecast: SRM Required Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
87	976.47	987.82	0.905100	9051
88	987.82	999.18	0.910200	9102
89	999.18	1,010.53	0.915100	9151
90	1,010.53	1,021.89	0.922400	9224
91	1,021.89	1,033.24	0.927300	9273
92	1,033.24	1,044.59	0.932400	9324
93	1,044.59	1,055.95	0.939800	9398
94	1,055.95	1,067.30	0.947300	9473
95	1,067.30	1,078.66	0.953700	9537
96	1,078.66	1,090.01	0.962600	9626
97	1,090.01	1,101.37	0.971200	9712
98	1,101.37	1,112.72	0.978500	9785
99	1,112.72	1,124.07	0.986900	9869
100	1,124.07	1,135.43	0.996400	9964
	1,135.43	+Infinity	1.000000	10000

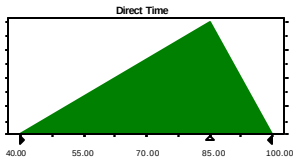
End of Forecast

Assumptions

Assumption: Direct Time **Cell: E8**

Triangular distribution with parameters:

Minimum	40.00
Likeliest	85.00
Maximum	100.00

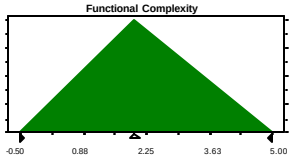


Selected range is from 40.00 to 100.00
Mean value in simulation was 74.99

Assumption: Functional Complexity **Cell: E11**

Triangular distribution with parameters:

Minimum	-0.50
Likeliest	2.00
Maximum	5.00

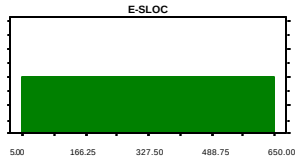


Selected range is from -0.50 to 5.00
Mean value in simulation was 2.19

Assumption: E-SLOC **Cell: E12**

Uniform distribution with parameters:

Minimum	5.00
Maximum	650.00

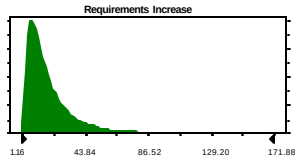


Mean value in simulation was 326.16

Assumption: Requirements Increase **Cell: E9**

Lognormal distribution with parameters:

Mean	20.00
Standard Dev.	20.00

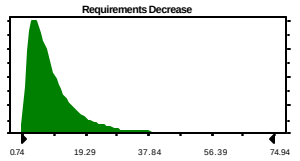


Selected range is from 0.00 to +Infinity
Mean value in simulation was 20.32

Assumption: Requirements Decrease **Cell: E10**

Lognormal distribution with parameters:

Mean	10.00
Standard Dev.	9.00

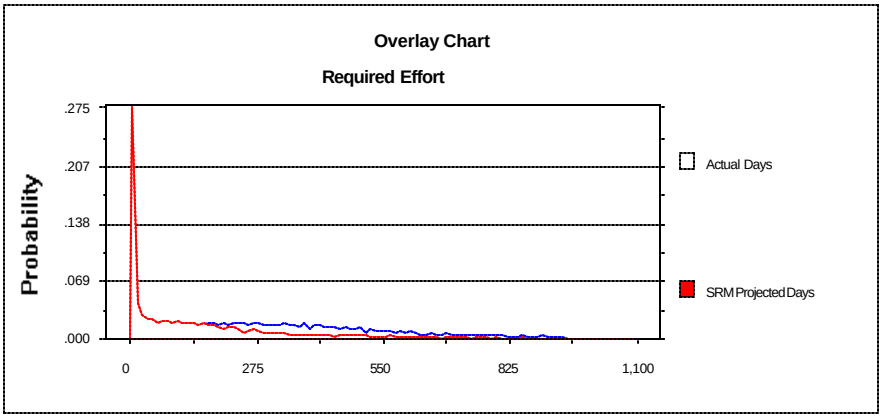


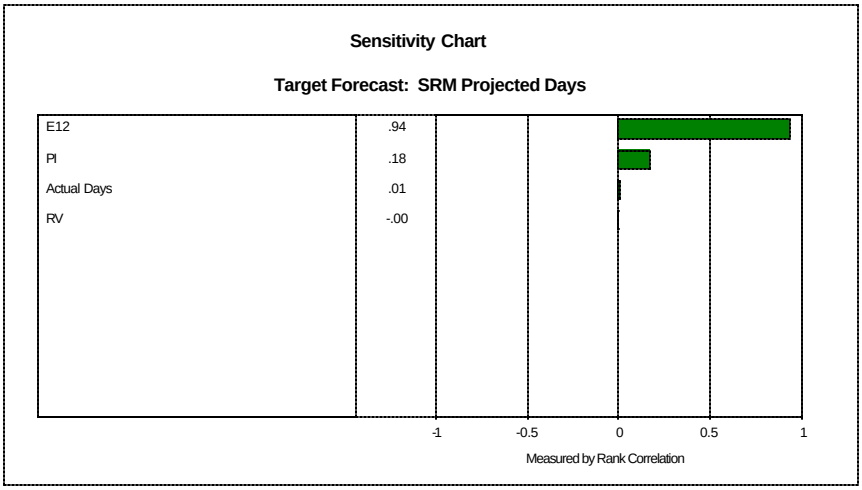
Selected range is from 0.00 to +Infinity
Mean value in simulation was 9.94

End of Assumptions

B. PROJECTIONS OF ACTUAL PROJECT DATA

Crystal Ball Report
Simulation started on 8/2/02 at 15:36:33
Simulation stopped on 8/2/02 at 15:37:52



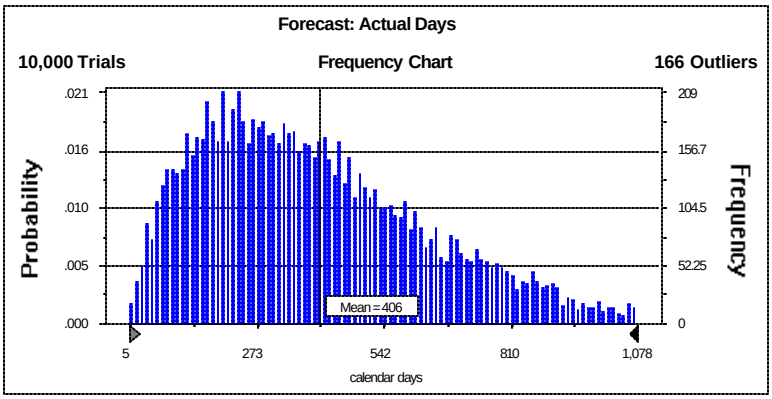


Forecast: Actual Days **Cell: L16**

Summary:
Display Range is from 5 to 1,078 calendar days

Entire Range is from 5 to 1,680 calendar days
After 10,000 Trials, the Std. Error of the Mean is 3

Statistics:	Value
Trials	10000
Mean	406
Median	362
Mode	---
Standard Deviation	254
Variance	64,572
Skewness	0.91
Kurtosis	3.73
Coeff. of Variability	0.63
Range Minimum	5
Range Maximum	1,680
Range Width	1,675
Mean Std. Error	2.54



Forecast: Actual Days (cont'd) **Cell: L16**

Percentiles:

Percentile	calendar days
0%	5
10%	118
20%	181
30%	238
40%	298
50%	362
60%	429
70%	505
80%	609

90%	762
100%	1,680

Frequency Counts:

Frequency:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	5	0.000000	0
1	5	16	0.001900	19
2	16	26	0.003700	37
3	26	37	0.005300	53
4	37	48	0.009100	91
5	48	59	0.007700	77
6	59	69	0.011000	110
7	69	80	0.012400	124
8	80	91	0.014000	140
9	91	102	0.014000	140
10	102	112	0.013500	135
11	112	123	0.013900	139
12	123	134	0.017100	171
13	134	145	0.015200	152
14	145	155	0.016800	168
15	155	166	0.016700	167
16	166	177	0.020100	201
17	177	187	0.018300	183
18	187	198	0.016600	166
19	198	209	0.020900	209
20	209	220	0.016400	164
21	220	230	0.019300	193
22	230	241	0.020900	209
23	241	252	0.018300	183
24	252	263	0.016300	163

Forecast: Actual Days (cont'd) **Cell: L16**

Group	Start Value	End Value	Prob.	Freq.
25	263	273	0.018600	186
26	273	284	0.017700	177
27	284	295	0.018300	183
28	295	306	0.017000	170
29	306	316	0.017100	171
30	316	327	0.016200	162
31	327	338	0.018100	181
32	338	348	0.017200	172
33	348	359	0.017500	175
34	359	370	0.015700	157

35	370	381	0.016200	162
36	381	391	0.016100	161
37	391	402	0.015100	151
38	402	413	0.016400	164
39	413	424	0.016900	169
40	424	434	0.014800	148
41	434	445	0.013400	134
42	445	456	0.016600	166
43	456	467	0.012700	127
44	467	477	0.015000	150
45	477	488	0.011400	114
46	488	499	0.013500	135
47	499	509	0.012300	123
48	509	520	0.011400	114
49	520	531	0.012200	122
50	531	542	0.010300	103
51	542	552	0.010600	106
52	552	563	0.010800	108
53	563	574	0.009800	98
54	574	585	0.009600	96
55	585	595	0.011000	110
56	595	606	0.008500	85
57	606	617	0.010100	101
58	617	628	0.008600	86
59	628	638	0.007000	70
60	638	649	0.007500	75
61	649	660	0.008800	88
62	660	670	0.006000	60
63	670	681	0.005600	56
64	681	692	0.008000	80
65	692	703	0.007700	77
66	703	713	0.006300	63

Forecast: Actual Days (cont'd) **Cell: L16**

Group	Start Value	End Value	Prob.	Freq.
67	713	724	0.005900	59
68	724	735	0.005500	55
69	735	746	0.006600	66
70	746	756	0.005800	58
71	756	767	0.005500	55
72	767	778	0.005100	51
73	778	789	0.005400	54
74	789	799	0.005100	51
75	799	810	0.004600	46
76	810	821	0.004400	44

77	821	831	0.003200	32
78	831	842	0.003800	38
79	842	853	0.003600	36
80	853	864	0.004800	48
81	864	874	0.003700	37
82	874	885	0.003300	33
83	885	896	0.003400	34
84	896	907	0.003600	36
85	907	917	0.003300	33
86	917	928	0.001600	16
87	928	939	0.002400	24
88	939	950	0.002200	22
89	950	960	0.001300	13
90	960	971	0.001900	19
91	971	982	0.001500	15
92	982	992	0.001500	15
93	992	1,003	0.002100	21
94	1,003	1,014	0.001200	12
95	1,014	1,025	0.001500	15
96	1,025	1,035	0.001400	14
97	1,035	1,046	0.001000	10
98	1,046	1,057	0.000900	9
99	1,057	1,068	0.001900	19
100	1,068	1,078	0.001400	14
	1,078	+Infinity	0.016600	166
	Total:		1.000000	10000

Cumulative:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	5	0.000000	0
1	5	16	0.001900	19
2	16	26	0.005600	56

Forecast: Actual Days (cont'd) **Cell: L16**

Group	Start Value	End Value	Prob.	Freq.
3	26	37	0.010900	109
4	37	48	0.020000	200
5	48	59	0.027700	277
6	59	69	0.038700	387
7	69	80	0.051100	511
8	80	91	0.065100	651
9	91	102	0.079100	791
10	102	112	0.092600	926
11	112	123	0.106500	1065
12	123	134	0.123600	1236
13	134	145	0.138800	1388

14	145	155	0.155600	1556
15	155	166	0.172300	1723
16	166	177	0.192400	1924
17	177	187	0.210700	2107
18	187	198	0.227300	2273
19	198	209	0.248200	2482
20	209	220	0.264600	2646
21	220	230	0.283900	2839
22	230	241	0.304800	3048
23	241	252	0.323100	3231
24	252	263	0.339400	3394
25	263	273	0.358000	3580
26	273	284	0.375700	3757
27	284	295	0.394000	3940
28	295	306	0.411000	4110
29	306	316	0.428100	4281
30	316	327	0.444300	4443
31	327	338	0.462400	4624
32	338	348	0.479600	4796
33	348	359	0.497100	4971
34	359	370	0.512800	5128
35	370	381	0.529000	5290
36	381	391	0.545100	5451
37	391	402	0.560200	5602
38	402	413	0.576600	5766
39	413	424	0.593500	5935
40	424	434	0.608300	6083
41	434	445	0.621700	6217
42	445	456	0.638300	6383
43	456	467	0.651000	6510
44	467	477	0.666000	6660

Forecast: Actual Days (cont'd) **Cell: L16**

Group	Start Value	End Value	Prob.	Freq.
45	477	488	0.677400	6774
46	488	499	0.690900	6909
47	499	509	0.703200	7032
48	509	520	0.714600	7146
49	520	531	0.726800	7268
50	531	542	0.737100	7371
51	542	552	0.747700	7477
52	552	563	0.758500	7585
53	563	574	0.768300	7683
54	574	585	0.777900	7779
55	585	595	0.788900	7889
56	595	606	0.797400	7974

57	606	617	0.807500	8075
58	617	628	0.816100	8161
59	628	638	0.823100	8231
60	638	649	0.830600	8306
61	649	660	0.839400	8394
62	660	670	0.845400	8454
63	670	681	0.851000	8510
64	681	692	0.859000	8590
65	692	703	0.866700	8667
66	703	713	0.873000	8730
67	713	724	0.878900	8789
68	724	735	0.884400	8844
69	735	746	0.891000	8910
70	746	756	0.896800	8968
71	756	767	0.902300	9023
72	767	778	0.907400	9074
73	778	789	0.912800	9128
74	789	799	0.917900	9179
75	799	810	0.922500	9225
76	810	821	0.926900	9269
77	821	831	0.930100	9301
78	831	842	0.933900	9339
79	842	853	0.937500	9375
80	853	864	0.942300	9423
81	864	874	0.946000	9460
82	874	885	0.949300	9493
83	885	896	0.952700	9527
84	896	907	0.956300	9563
85	907	917	0.959600	9596
86	917	928	0.961200	9612

Forecast: Actual Days (cont'd) **Cell: L16**

Group	Start Value	End Value	Prob.	Freq.
87	928	939	0.963600	9636
88	939	950	0.965800	9658
89	950	960	0.967100	9671
90	960	971	0.969000	9690
91	971	982	0.970500	9705
92	982	992	0.972000	9720
93	992	1,003	0.974100	9741
94	1,003	1,014	0.975300	9753
95	1,014	1,025	0.976800	9768
96	1,025	1,035	0.978200	9782
97	1,035	1,046	0.979200	9792
98	1,046	1,057	0.980100	9801
99	1,057	1,068	0.982000	9820

100	1,068	1,078	0.983400	9834
	1,078	+Infinity	1.000000	10000

End of Forecast

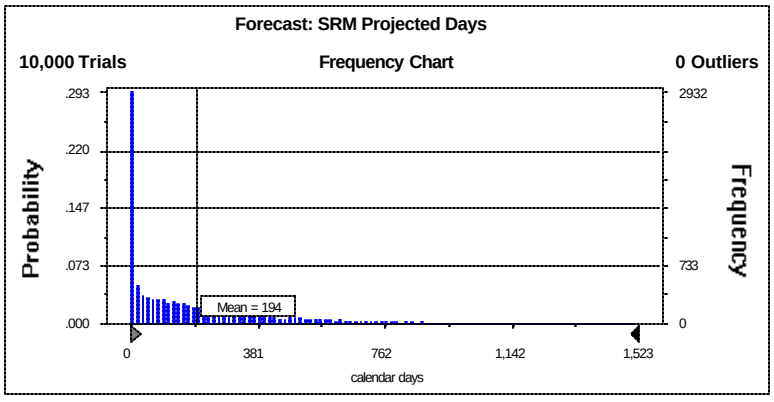
Cell: L14

Forecast: SRM Projected Days

Summary:

Display Range is from 0 to 1,523 calendar days
Entire Range is from 0 to 1,523 calendar days
After 10,000 Trials, the Std. Error of the Mean is 2

Statistics:	Value
Trials	10000
Mean	194
Median	106
Mode	0
Standard Deviation	243
Variance	59,133
Skewness	1.73
Kurtosis	6.03
Coeff. of Variability	1.25
Range Minimum	0
Range Maximum	1,523
Range Width	1,523
Mean Std. Error	2.43



Forecast: SRM Projected Days (cont'd) **Cell: L14**

Percentiles:

Percentile	calendar days
0%	0
10%	0

20%	2
30%	17
40%	57
50%	106
60%	162
70%	233
80%	343
90%	546
100%	1,523

Frequency Counts:

Frequency:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	0	0.000000	0
1	0	15	0.293200	2932
2	15	30	0.047500	475
3	30	46	0.034400	344
4	46	61	0.033700	337
5	61	76	0.030100	301
6	76	91	0.031700	317
7	91	107	0.030000	300
8	107	122	0.027800	278
9	122	137	0.029000	290
10	137	152	0.025500	255
11	152	168	0.027000	270
12	168	183	0.022900	229
13	183	198	0.021000	210
14	198	213	0.020400	204
15	213	228	0.020500	205
16	228	244	0.018000	180
17	244	259	0.014000	140
18	259	274	0.016300	163
19	274	289	0.014900	149
20	289	305	0.011800	118
21	305	320	0.012700	127
22	320	335	0.011800	118
23	335	350	0.010400	104
24	350	366	0.009800	98

Forecast: SRM Projected Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
25	366	381	0.009700	97
26	381	396	0.010100	101
27	396	411	0.008600	86

28	411	426	0.007700	77
29	426	442	0.006800	68
30	442	457	0.006100	61
31	457	472	0.006000	60
32	472	487	0.007300	73
33	487	503	0.006500	65
34	503	518	0.007200	72
35	518	533	0.005500	55
36	533	548	0.004500	45
37	548	564	0.005200	52
38	564	579	0.005200	52
39	579	594	0.005000	50
40	594	609	0.004600	46
41	609	625	0.003200	32
42	625	640	0.004700	47
43	640	655	0.004100	41
44	655	670	0.003800	38
45	670	685	0.003100	31
46	685	701	0.003300	33
47	701	716	0.004100	41
48	716	731	0.003200	32
49	731	746	0.003000	30
50	746	762	0.003200	32
51	762	777	0.003400	34
52	777	792	0.002200	22
53	792	807	0.003300	33
54	807	823	0.002000	20
55	823	838	0.002700	27
56	838	853	0.002500	25
57	853	868	0.002100	21
58	868	883	0.002300	23
59	883	899	0.001800	18
60	899	914	0.001400	14
61	914	929	0.002000	20
62	929	944	0.001800	18
63	944	960	0.001600	16
64	960	975	0.001200	12
65	975	990	0.001300	13
66	990	1,005	0.001200	12

Forecast: SRM Projected Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
67	1,005	1,021	0.000600	6
68	1,021	1,036	0.000800	8
69	1,036	1,051	0.000900	9
70	1,051	1,066	0.001000	10

71	1,066	1,081	0.000900	9
72	1,081	1,097	0.000700	7
73	1,097	1,112	0.000200	2
74	1,112	1,127	0.000700	7
75	1,127	1,142	0.000200	2
76	1,142	1,158	0.000600	6
77	1,158	1,173	0.000600	6
78	1,173	1,188	0.000200	2
79	1,188	1,203	0.000400	4
80	1,203	1,219	0.000500	5
81	1,219	1,234	0.000400	4
82	1,234	1,249	0.000000	0
83	1,249	1,264	0.000200	2
84	1,264	1,279	0.000500	5
85	1,279	1,295	0.000200	2
86	1,295	1,310	0.000100	1
87	1,310	1,325	0.000100	1
88	1,325	1,340	0.000000	0
89	1,340	1,356	0.000300	3
90	1,356	1,371	0.000100	1
91	1,371	1,386	0.000100	1
92	1,386	1,401	0.000000	0
93	1,401	1,417	0.000100	1
94	1,417	1,432	0.000200	2
95	1,432	1,447	0.000100	1
96	1,447	1,462	0.000100	1
97	1,462	1,478	0.000000	0
98	1,478	1,493	0.000100	1
99	1,493	1,508	0.000100	1
100	1,508	1,523	0.000100	1
	1,523	+Infinity	0.000000	0
Total:			1.000000	10000

Cumulative:

Group	Start Value	End Value	Prob.	Freq.
	-Infinity	0	0.000000	0
1	0	15	0.293200	2932
2	15	30	0.340700	3407

Forecast: SRM Projected Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
3	30	46	0.375100	3751
4	46	61	0.408800	4088
5	61	76	0.438900	4389
6	76	91	0.470600	4706
7	91	107	0.500600	5006

8	107	122	0.528400	5284
9	122	137	0.557400	5574
10	137	152	0.582900	5829
11	152	168	0.609900	6099
12	168	183	0.632800	6328
13	183	198	0.653800	6538
14	198	213	0.674200	6742
15	213	228	0.694700	6947
16	228	244	0.712700	7127
17	244	259	0.726700	7267
18	259	274	0.743000	7430
19	274	289	0.757900	7579
20	289	305	0.769700	7697
21	305	320	0.782400	7824
22	320	335	0.794200	7942
23	335	350	0.804600	8046
24	350	366	0.814400	8144
25	366	381	0.824100	8241
26	381	396	0.834200	8342
27	396	411	0.842800	8428
28	411	426	0.850500	8505
29	426	442	0.857300	8573
30	442	457	0.863400	8634
31	457	472	0.869400	8694
32	472	487	0.876700	8767
33	487	503	0.883200	8832
34	503	518	0.890400	8904
35	518	533	0.895900	8959
36	533	548	0.900400	9004
37	548	564	0.905600	9056
38	564	579	0.910800	9108
39	579	594	0.915800	9158
40	594	609	0.920400	9204
41	609	625	0.923600	9236
42	625	640	0.928300	9283
43	640	655	0.932400	9324
44	655	670	0.936200	9362

Forecast: SRM Projected Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
45	670	685	0.939300	9393
46	685	701	0.942600	9426
47	701	716	0.946700	9467
48	716	731	0.949900	9499
49	731	746	0.952900	9529

50	746	762	0.956100	9561
51	762	777	0.959500	9595
52	777	792	0.961700	9617
53	792	807	0.965000	9650
54	807	823	0.967000	9670
55	823	838	0.969700	9697
56	838	853	0.972200	9722
57	853	868	0.974300	9743
58	868	883	0.976600	9766
59	883	899	0.978400	9784
60	899	914	0.979800	9798
61	914	929	0.981800	9818
62	929	944	0.983600	9836
63	944	960	0.985200	9852
64	960	975	0.986400	9864
65	975	990	0.987700	9877
66	990	1,005	0.988900	9889
67	1,005	1,021	0.989500	9895
68	1,021	1,036	0.990300	9903
69	1,036	1,051	0.991200	9912
70	1,051	1,066	0.992200	9922
71	1,066	1,081	0.993100	9931
72	1,081	1,097	0.993800	9938
73	1,097	1,112	0.994000	9940
74	1,112	1,127	0.994700	9947
75	1,127	1,142	0.994900	9949
76	1,142	1,158	0.995500	9955
77	1,158	1,173	0.996100	9961
78	1,173	1,188	0.996300	9963
79	1,188	1,203	0.996700	9967
80	1,203	1,219	0.997200	9972
81	1,219	1,234	0.997600	9976
82	1,234	1,249	0.997600	9976
83	1,249	1,264	0.997800	9978
84	1,264	1,279	0.998300	9983
85	1,279	1,295	0.998500	9985
86	1,295	1,310	0.998600	9986

Forecast: SRM Projected Days (cont'd) **Cell: L14**

Group	Start Value	End Value	Prob.	Freq.
87	1,310	1,325	0.998700	9987
88	1,325	1,340	0.998700	9987
89	1,340	1,356	0.999000	9990
90	1,356	1,371	0.999100	9991
91	1,371	1,386	0.999200	9992
92	1,386	1,401	0.999200	9992

93	1,401	1,417	0.999300	9993
94	1,417	1,432	0.999500	9995
95	1,432	1,447	0.999600	9996
96	1,447	1,462	0.999700	9997
97	1,462	1,478	0.999700	9997
98	1,478	1,493	0.999800	9998
99	1,493	1,508	0.999900	9999
100	1,508	1,523	1.000000	10000
	1,523	+Infinity	1.000000	10000

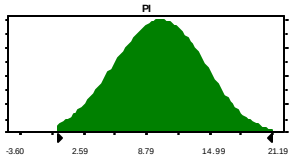
End of Forecast

Assumptions

Assumption: PI **Cell: E6**

Weibull distribution with parameters:

Location	-3.60
Scale	15.11
Shape	3.615131262

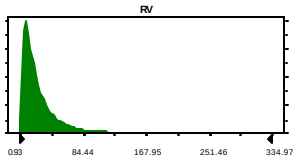


Selected range is from 0.00 to +Infinity
Mean value in simulation was 10.08

Assumption: RV **Cell: E5**

Lognormal distribution with parameters:

Mean	28.59
Standard Dev.	36.34

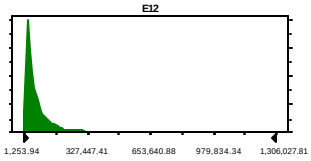


Selected range is from 0.00 to +Infinity
Mean value in simulation was 28.37

Assumption: E12 **Cell: E12**

Lognormal distribution with parameters:

Mean	79,129.70
Standard Dev.	132,961.36

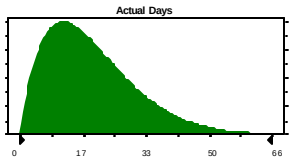


Selected range is from 0.00 to +Infinity
Mean value in simulation was 78,611.39

Assumption: Actual Days **Cell: L15**

Beta distribution with parameters:

Alpha	1.98
Beta	9.80
Scale	111



Selected range is from 0 to +Infinity
Mean value in simulation was 18

End of Assumptions

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[Bark93] Barki, H., Rivard S., and Talbot J., *Toward an Assessment of Software Development Risk*. J. Management Information Systems, Vol. 10, No. 2, 1993, pp. 203-225.

[Boeh81] Boehm, B. *Software Engineering Economics.* Prentice Hall, 1981.

[Carr93] M. J. Carr et al., *Taxonomy-Based Risk Identification*, SEI-93-TR-006, Software Eng. Inst., Pittsburgh, 1993.

[Crys93] Crystal Ball® version 3.0: User's Manual/Decisioneering, Inc. 1993.

[Dupo02] Dupont, Joseph *Complexity Measure for the Prototype System Description Language (PSDL).* Master's Thesis. Naval Postgraduate School. Monterey, California. June 2002.

[Fent00] Fenton, N. E. and Neil, M. *Software Metrics: Roadmap.* Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 357- 370.

[Garv97] Garvey, P. R., Phair, D. J., Wilson, J. A, *An Information Architecture For Risk Assessment And Management.* IEEE Software, Volume: 14 Issue: 3, May-June 1997, pp. 25–34.

[IEEE01] IEEE Std 1540-2001. Software Engineering Standards Committee of the IEEE Computer Society. Approved March 17, 2001.

[John01] Johnson, C. S., Piirainen R. A., *Application of the Nogueira Risk Assessment Model to Real-Time Embedded Software Projects.* Master's Thesis. Naval Postgraduate School. Monterey, California. March 2001.

[Kesh00] Keshlaf, A. and Hashim, K. *A Model and Prototype Tool to Manage Software Risks.* Quality Software, 2000. Proceedings. First Asia-Pacific Conference on, 2000. pp. 297–305.

[Luqi90] Berzins, V. and Luqi. *Software Engineering with Abstractions.* Addison-Wesley, 1990.

[Moyn97] Moynihan, T., *How Experienced Project Managers Assess Risk.* IEEE Software, Volume: 14 Issue: 3, May-June 1997. Page(s): 35–41.

[Nogu00] Nogueira J. C., *A Formal Model for Risk Assessment in Software Projects.* PhD Dissertation. Naval Postgraduate School. Monterey, California, 2000.

[Putn92] Putnam, L. and Myers, W. *Measures for Excellence. Reliable Software On Time Within Budget.* Yourdon Press, 1992.

[QSMa] Quantitative Software Management®. *User's Guide to SLIM-Metrics 5.0.* QSM, Inc., McLean, Virginia, USA.

[QSMb] Quantitative Software Management®. *User's Guide to SLIM-Estimate 5.0.* QSM, Inc. McLean, Virginia, USA.

[QSMc] Quantitative Software Management®. *Software Lifecycle Management (SLIM) Training Version 2.02.* QSM, Inc. McLean, Virginia, USA.

[Thay81] Richard H. Thayer, Arthur B. Pyster, Roger C. Wood: Major Issues in Software Engineering Project Management. TSE 7(4): 333-342 (1981)

BIBLIOGRAPHY

Abdel-Hamid, T. *Software Project Dynamics: An Integrated Approach.* Prentice Hall, 1991.

ANSI/IEEE *Standard Glossary of Software Engineering Terminology*. STD-729-1991.

Albrecht, A. and Gaffney, J. *Software Function Source Lines of Code and Development Effort prediction.* IEEE Transactions on Software Engineering, SE-9, 1983.

Berzins, V. and Luqi. *Software Engineering with Abstractions*. Addison-Wesley, 1990.

Boehm, B. *A Spiral Model of Software Development and Enhancement. Computer*. May, 1988.

Boehm, B. *Software Risk Management: Principles and Practices.* IEEE Software, January, 1991.

Boehm, B. et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

Brooks, F. *The Mythical Man-Month.* Datamation. December. 1974.

Charette, R., Adams, K., & White, M. *Managing Risk in Software Maintenance*. IEEE Software, May-June, 1997.

Gemmer. *Risk Management: Moving Beyond Process*. Computer Vol. 30 Issue 5. May, 1997.

Gilb, T. *Principles of Software Engineering Management.* Addison-Wesley 1988.

Harn, M., Berzins, V. and Luqi. *Software Evolution via Reusable Architecture.* Proceedings of 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems. Nashville, Tennessee. March, 1999.

Humphrey, W. et al. *A Method for Assessing the Software Capability of Contractors.* CMU/SEI-87-TR-23. 1987.

Humphrey, W. *Managing the Software Process.* Addison-Wesley, 1989.

Jones, Capers. *By Popular Demand: Software Estimating Rules of Thumb*. Computer, March 1996.

Luqi and Ketabchi, M. *A Computer-Aided Prototyping System.* IEEE Software. March, 1988.

Luqi and Berzins, V. *Rapidly Prototyping Real-Time Systems.* IEEE Software. September, 1988.

Luqi. Software Evolution *Through Rapid Prototyping.* IEEE Computer. May, 1989.

Luqi. *A Graph Model for Software Evolution.* IEEE Transactions on Software Engineering. Vol. 16 No. 8. August, 1990.

Nogueira, J. C., Luqi, and Berzins, V. *A Formal Risk Assessment Model for Software Evolution.* SEKE 2000. Chicago, July 2000.

Nogueira, J. C., Luqi, and Bhattacharya, S. *A Risk Assessment Model for Software Prototyping Projects.* IEEE Workshop on Rapid System Prototyping RSP 2000. Paris, June 2000.

Nogueira, J. C., Luqi, , Berzins, V., and Nada, N. *A Formal Risk Assessment Model for Software Evolution.* ICSE 2000. Limerick, June 2000.

Nogueira, J. C., Luqi, and Berzins, V. *Risk Assessment in Software Requirement Engineering.* IDTP 2000. Dallas, June 2000.

Parametric Estimating Handbook, Spring 1999, Department of Defense.

Putnam, L. *Software Cost Estimating and Life-cycle Control: Getting the Software Numbers.* IEEE Computer Society Press. 1980.

Putnam, L. and Myers, W. *Measures for Excellence. Reliable Software On Time Within Budget.* Yourdon Press, 1992.

Software Engineering Institute. *Software Risk Management*. Technical Report CMU/SEI-96-TR-012. June, 1996.

USAF. *Software Risk Abatement.* ASFC/AFLC pamphlet 800-45, US Air Force Systems Command. Andrews AFB. 1988.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Luqi
   Naval Postgraduate School
   Code CS/Lq
   Monterey, California

4. Major Michael R. Murrah
   Army Research Lab
   Adelphi, Maryland